

# **DRONE\_SOC**

**Application Processor**

**Datasheet**

**October 23, 2018**

# Contents

Contents . . . . .	2
<b>1 Product Overview</b>	<b>4</b>
1.1 Introduction . . . . .	4
1.2 Features . . . . .	4
1.3 Block Diagram . . . . .	5
1.4 Brief Functional Specification . . . . .	6
1.4.1 SYSREG . . . . .	6
1.4.2 PLL . . . . .	6
1.4.3 CMU . . . . .	6
1.4.4 USB20OTG . . . . .	6
1.4.5 DDRC . . . . .	7
1.4.6 HOVER_DMA . . . . .	7
1.4.7 VIP . . . . .	7
1.4.8 Multi-Format Video Codec . . . . .	7
1.4.9 Scaler . . . . .	8
1.4.10 Crypto Engine . . . . .	8
1.4.11 ADC . . . . .	9
1.4.12 I2C . . . . .	9
1.4.13 SD/MMC Controller . . . . .	9
1.4.14 DMA . . . . .	10
1.4.15 GPIO . . . . .	10
1.4.16 UART & ISO7816 Sim Card Interface . . . . .	11
1.4.17 PWM . . . . .	12
1.4.18 WDT . . . . .	12
<b>2 Mechanical Dimension</b>	<b>13</b>

2.1	Mechanical Dimension . . . . .	13
<b>3</b>	<b>Address Map</b>	<b>14</b>
3.1	Overview . . . . .	14
3.2	SFR Address Map . . . . .	15
<b>4</b>	<b>SYSREG</b>	<b>17</b>
4.1	Overview . . . . .	17
4.2	Features . . . . .	17
4.3	SYSREG_SYS Register Description . . . . .	18
4.3.1	Register Map Summary . . . . .	18
4.3.2	SPARE0 . . . . .	19
4.3.3	SPARE1 . . . . .	19
4.3.4	SPARE2 . . . . .	19
4.3.5	SPARE3 . . . . .	19
4.3.6	SPARE4 . . . . .	20
4.3.7	HOVER_0_CTRL0 . . . . .	20
4.3.8	RSP_0_CTRL0 . . . . .	20
4.3.9	DDRC_0_CTRL0 . . . . .	20
4.3.10	SPI_0_CTRL0 . . . . .	21
4.3.11	SPI_1_CTRL0 . . . . .	21
4.3.12	SPI_2_CTRL0 . . . . .	21
4.3.13	USART_0_CTRL0 . . . . .	21
4.3.14	USART_1_CTRL0 . . . . .	22
4.3.15	USART_2_CTRL0 . . . . .	22
4.3.16	USART_3_CTRL0 . . . . .	22
4.3.17	PWM_0_CTRL0 . . . . .	22
4.3.18	PWM_1_CTRL0 . . . . .	23
4.3.19	PWM_2_CTRL0 . . . . .	23
4.3.20	TIMER_0_CTRL0 . . . . .	23

4.3.21	TIMER_1_CTRL0 . . . . .	24
4.3.22	USB_PHY_CFG_0 . . . . .	24
4.3.23	USB_PHY_CFG_1 . . . . .	24
4.3.24	USB_PHY_CFG_2 . . . . .	25
4.3.25	DMA_BOOT_CFG . . . . .	25
4.3.26	DMA_0_CTRL0 . . . . .	25
4.3.27	DMA_1_CTRL0 . . . . .	26
4.3.28	DMA_0_CTRL0 . . . . .	26
4.3.29	DMA_0_CTRL0 . . . . .	26
4.3.30	USB_PHY_CFG_3 . . . . .	26
<b>5</b>	<b>PLL</b>	<b>27</b>
5.1	Overview . . . . .	27
5.2	Features . . . . .	27
5.3	Block Diagram . . . . .	28
5.4	PLL Functional Description . . . . .	29
5.5	PLL Register Description . . . . .	30
5.5.1	Register Map Summary . . . . .	30
5.5.2	PLLCTRL . . . . .	31
5.5.3	PLLDDBG0 . . . . .	31
5.5.4	PLLCNT0 . . . . .	31
5.5.5	PLLCNT1 . . . . .	32
5.5.6	PLLCNT2 . . . . .	32
5.5.7	PLLCNT3 . . . . .	32
5.5.8	PLLCFG0 . . . . .	32
5.5.9	PLLCFG1 . . . . .	33
5.5.10	PLLLOCKINT0 . . . . .	33
<b>6</b>	<b>CMU</b>	<b>34</b>
6.1	Overview . . . . .	34

6.2	Features . . . . .	34
6.3	Application Guide . . . . .	35
6.3.1	S/W Reset Release Guide . . . . .	35
6.4	Clock Source Description . . . . .	35
6.5	CMU_SYS . . . . .	36
6.5.1	SYS_0_CLK400_CLK . . . . .	37
6.5.1.1	Block Diagram . . . . .	37
6.5.1.2	Register Map Summary . . . . .	37
6.5.1.3	CLKMUX_SYS_0_CLK400_CLK . . . . .	40
6.5.1.4	GRPRST_SYS_0_CLK400_CLK . . . . .	40
6.5.1.5	GATEMODE_SYS_0_CLK400_CLK . . . . .	40
6.5.1.6	CLKENB_SYS_0_CLK400_CLK0 . . . . .	40
6.5.1.7	CLKENB_SYS_0_CLK400_CLK1 . . . . .	41
6.5.1.8	CLKENB_SYS_0_CLK400_CLK2 . . . . .	43
6.5.1.9	CLKENB_SYS_0_CLK400_CLK3 . . . . .	44
6.5.1.10	DIVRST_SYS_0_CLK400_CLK . . . . .	44
6.5.1.11	DIVVAL_SYS_0_CLK400_CLK . . . . .	45
6.5.1.12	STOP_AND_GO_RESET_SYS_0_CLK400_CLK . . . . .	45
6.5.1.13	DIVSTATUS_SYS_0_CLK400_CLK . . . . .	45
6.5.1.14	DIVRST_SYS_0_AXI_CLK . . . . .	45
6.5.1.15	DIVVAL_SYS_0_AXI_CLK . . . . .	46
6.5.1.16	STOP_AND_GO_RESET_SYS_0_AXI_CLK . . . . .	46
6.5.1.17	DIVSTATUS_SYS_0_AXI_CLK . . . . .	46
6.5.1.18	DIVRST_SYS_0_APB_CLK . . . . .	46
6.5.1.19	DIVVAL_SYS_0_APB_CLK . . . . .	47
6.5.1.20	STOP_AND_GO_RESET_SYS_0_APB_CLK . . . . .	47
6.5.1.21	DIVSTATUS_SYS_0_APB_CLK . . . . .	47
6.5.1.22	DIVRST_SYS_0_CLK133_CLK . . . . .	47
6.5.1.23	DIVVAL_SYS_0_CLK133_CLK . . . . .	47

6.5.1.24	STOP_AND_GO_RESET_SYS_0_CLK133_CLK . . . . .	48
6.5.1.25	DIVSTATUS_SYS_0_CLK133_CLK . . . . .	48
6.5.1.26	DIVRST_SYS_0_CLK50_CLK . . . . .	48
6.5.1.27	DIVVAL_SYS_0_CLK50_CLK . . . . .	48
6.5.1.28	STOP_AND_GO_RESET_SYS_0_CLK50_CLK . . . . .	49
6.5.1.29	DIVSTATUS_SYS_0_CLK50_CLK . . . . .	49
6.5.1.30	DIVRST_SYS_0_CLK40_CLK . . . . .	49
6.5.1.31	DIVVAL_SYS_0_CLK40_CLK . . . . .	49
6.5.1.32	STOP_AND_GO_RESET_SYS_0_CLK40_CLK . . . . .	50
6.5.1.33	DIVSTATUS_SYS_0_CLK40_CLK . . . . .	50
6.5.2	CPU_0_CORE_CLK . . . . .	51
6.5.2.1	Block Diagram . . . . .	51
6.5.2.2	Register Map Summary . . . . .	51
6.5.2.3	CLKMUX_CPU_0_CORE_CLK . . . . .	52
6.5.2.4	GRPRST_CPU_0_CORE_CLK . . . . .	52
6.5.2.5	GATEMODE_CPU_0_CORE_CLK . . . . .	52
6.5.2.6	CLKENB_CPU_0_CORE_CLK0 . . . . .	52
6.5.2.7	DIVRST_CPU_0_CORE_CLK . . . . .	52
6.5.2.8	DIVVAL_CPU_0_CORE_CLK . . . . .	53
6.5.2.9	STOP_AND_GO_RESET_CPU_0_CORE_CLK . . . . .	53
6.5.2.10	DIVSTATUS_CPU_0_CORE_CLK . . . . .	53
<b>7</b>	<b>USB2.0 DEVICE</b> . . . . .	<b>54</b>
7.1	Features . . . . .	54
7.2	Block Diagram . . . . .	55
7.3	Functional Description . . . . .	56
7.3.1	End Point Packet Size . . . . .	56
7.3.2	Modes of Operation . . . . .	56
7.3.3	Programming User Config of PHY and LINK . . . . .	56

7.4	USB2.0 DEVICE Controller Register Description . . . . .	57
7.4.1	Register Map Summary . . . . .	57
7.4.2	GOTGCTL . . . . .	62
7.4.3	GOTGINT . . . . .	65
7.4.4	GAHBCFG . . . . .	66
7.4.5	GUSBCFG . . . . .	68
7.4.6	GRSTCTL . . . . .	71
7.4.7	GINTSTS . . . . .	73
7.4.8	GINTMSK . . . . .	77
7.4.9	GRXSTSR . . . . .	78
7.4.10	GRXSTSP . . . . .	79
7.4.11	GRXFSIZ . . . . .	80
7.4.12	GNPTXFSIZ . . . . .	80
7.4.13	GNPTXSTS . . . . .	81
7.4.14	GGPIO . . . . .	81
7.4.15	GUID . . . . .	82
7.4.16	GSNPSID . . . . .	82
7.4.17	GHWCFG1 . . . . .	82
7.4.18	GHWCFG2 . . . . .	83
7.4.19	GHWCFG3 . . . . .	84
7.4.20	GHWCFG4 . . . . .	85
7.4.21	GLPMCFG . . . . .	86
7.4.22	GPWRDN . . . . .	90
7.4.23	GDFIFO CFG . . . . .	92
7.4.24	GADPCTL . . . . .	92
7.4.25	HPTXFSIZ . . . . .	94
7.4.26	DIEPTXF1 . . . . .	94
7.4.27	DIEPTXF2 . . . . .	94
7.4.28	DIEPTXF3 . . . . .	95

7.4.29	DIEPTXF4 . . . . .	95
7.4.30	DIEPTXF5 . . . . .	96
7.4.31	DIEPTXF6 . . . . .	96
7.4.32	DIEPTXF7 . . . . .	96
7.4.33	DIEPTXF8 . . . . .	97
7.4.34	DIEPTXF9 . . . . .	97
7.4.35	DIEPTXF10 . . . . .	98
7.4.36	DIEPTXF11 . . . . .	98
7.4.37	DIEPTXF12 . . . . .	98
7.4.38	DIEPTXF13 . . . . .	99
7.4.39	DIEPTXF14 . . . . .	99
7.4.40	DIEPTXF15 . . . . .	100
7.4.41	DCFG . . . . .	100
7.4.42	DCTL . . . . .	102
7.4.43	DSTS . . . . .	105
7.4.44	DIEPMSK . . . . .	105
7.4.45	DOEPMSK . . . . .	106
7.4.46	DAINT . . . . .	106
7.4.47	DAINTMSK . . . . .	107
7.4.48	DVBUSDIS . . . . .	108
7.4.49	DVBUSPULSE . . . . .	108
7.4.50	DTHRCTL . . . . .	109
7.4.51	DIEPEMPMSK . . . . .	110
7.4.52	DIEPCTL0 . . . . .	111
7.4.53	DIEPINT0 . . . . .	112
7.4.54	DIEPTSIZ0 . . . . .	114
7.4.55	DIEPDMA0 . . . . .	114
7.4.56	DTXFSTS0 . . . . .	115
7.4.57	DIEPDMA0 . . . . .	115

7.4.58	DIEPCTL1 . . . . .	116
7.4.59	DIEPINT1 . . . . .	119
7.4.60	DIEPTSIZ1 . . . . .	120
7.4.61	DIEPDMA1 . . . . .	121
7.4.62	DTXFSTS1 . . . . .	121
7.4.63	DIEPDMAB1 . . . . .	121
7.4.64	DIEPCTL2 . . . . .	122
7.4.65	DIEPINT2 . . . . .	125
7.4.66	DIEPTSIZ2 . . . . .	126
7.4.67	DIEPDMA2 . . . . .	127
7.4.68	DTXFSTS2 . . . . .	127
7.4.69	DIEPDMAB2 . . . . .	127
7.4.70	DIEPCTL3 . . . . .	128
7.4.71	DIEPINT3 . . . . .	131
7.4.72	DIEPTSIZ3 . . . . .	132
7.4.73	DIEPDMA3 . . . . .	133
7.4.74	DTXFSTS3 . . . . .	133
7.4.75	DIEPDMAB3 . . . . .	133
7.4.76	DIEPCTL4 . . . . .	134
7.4.77	DIEPINT4 . . . . .	137
7.4.78	DIEPTSIZ4 . . . . .	138
7.4.79	DIEPDMA4 . . . . .	139
7.4.80	DTXFSTS4 . . . . .	139
7.4.81	DIEPDMAB4 . . . . .	139
7.4.82	DIEPCTL5 . . . . .	140
7.4.83	DIEPINT5 . . . . .	143
7.4.84	DIEPTSIZ5 . . . . .	144
7.4.85	DIEPDMA5 . . . . .	145
7.4.86	DTXFSTS5 . . . . .	145

7.4.87	DIEPDMAB5 . . . . .	145
7.4.88	DIEPCTL6 . . . . .	146
7.4.89	DIEPINT6 . . . . .	149
7.4.90	DIEPTSIZ6 . . . . .	150
7.4.91	DIEPDMA6 . . . . .	151
7.4.92	DTXFSTS6 . . . . .	151
7.4.93	DIEPDMAB6 . . . . .	151
7.4.94	DIEPCTL7 . . . . .	152
7.4.95	DIEPINT7 . . . . .	155
7.4.96	DIEPTSIZ7 . . . . .	156
7.4.97	DIEPDMA7 . . . . .	157
7.4.98	DTXFSTS7 . . . . .	157
7.4.99	DIEPDMAB7 . . . . .	157
7.4.100	DIEPCTL8 . . . . .	158
7.4.101	DIEPINT8 . . . . .	161
7.4.102	DIEPTSIZ8 . . . . .	162
7.4.103	DIEPDMA8 . . . . .	163
7.4.104	DTXFSTS8 . . . . .	163
7.4.105	DIEPDMAB8 . . . . .	163
7.4.106	DIEPCTL9 . . . . .	164
7.4.107	DIEPINT9 . . . . .	167
7.4.108	DIEPTSIZ9 . . . . .	168
7.4.109	DIEPDMA9 . . . . .	169
7.4.110	DTXFSTS9 . . . . .	169
7.4.111	DIEPDMAB9 . . . . .	169
7.4.112	DIEPCTL10 . . . . .	170
7.4.113	DIEPINT10 . . . . .	173
7.4.114	DIEPTSIZ10 . . . . .	174
7.4.115	DIEPDMA10 . . . . .	175

7.4.116	DTXFSTS10 . . . . .	175
7.4.117	DIEPDMAB10 . . . . .	175
7.4.118	DIEPCTL11 . . . . .	176
7.4.119	DIEPINT11 . . . . .	179
7.4.120	DIEPTSIZ11 . . . . .	180
7.4.121	DIEPDMA11 . . . . .	181
7.4.122	DTXFSTS11 . . . . .	181
7.4.123	DIEPDMAB11 . . . . .	181
7.4.124	DIEPCTL12 . . . . .	182
7.4.125	DIEPINT12 . . . . .	185
7.4.126	DIEPTSIZ12 . . . . .	186
7.4.127	DIEPDMA12 . . . . .	187
7.4.128	DTXFSTS12 . . . . .	187
7.4.129	DIEPDMAB12 . . . . .	187
7.4.130	DIEPCTL13 . . . . .	188
7.4.131	DIEPINT13 . . . . .	191
7.4.132	DIEPTSIZ13 . . . . .	192
7.4.133	DIEPDMA13 . . . . .	193
7.4.134	DTXFSTS13 . . . . .	193
7.4.135	DIEPDMAB13 . . . . .	193
7.4.136	DIEPCTL14 . . . . .	194
7.4.137	DIEPINT14 . . . . .	197
7.4.138	DIEPTSIZ14 . . . . .	198
7.4.139	DIEPDMA14 . . . . .	199
7.4.140	DTXFSTS14 . . . . .	199
7.4.141	DIEPDMAB14 . . . . .	199
7.4.142	DIEPCTL15 . . . . .	200
7.4.143	DIEPINT15 . . . . .	203
7.4.144	DIEPTSIZ15 . . . . .	204

7.4.145	DIEPDMA15 . . . . .	205
7.4.146	DTXFSTS15 . . . . .	205
7.4.147	DIEPDMAB15 . . . . .	205
7.4.148	DOEPCTL0 . . . . .	206
7.4.149	DOEPINT0 . . . . .	207
7.4.150	DOEPTSIZ0 . . . . .	208
7.4.151	DOEPDMA0 . . . . .	209
7.4.152	DOEPDMAB0 . . . . .	209
7.4.153	DOEPCTL1 . . . . .	210
7.4.154	DOEPINT1 . . . . .	213
7.4.155	DOEPTSIZ1 . . . . .	214
7.4.156	DOEPDMA1 . . . . .	215
7.4.157	DOEPDMAB1 . . . . .	215
7.4.158	DOEPCTL2 . . . . .	216
7.4.159	DOEPINT2 . . . . .	219
7.4.160	DOEPTSIZ2 . . . . .	220
7.4.161	DOEPDMA2 . . . . .	221
7.4.162	DOEPDMAB2 . . . . .	221
7.4.163	DOEPCTL3 . . . . .	222
7.4.164	DOEPINT3 . . . . .	225
7.4.165	DOEPTSIZ3 . . . . .	226
7.4.166	DOEPDMA3 . . . . .	227
7.4.167	DOEPDMAB3 . . . . .	227
7.4.168	DOEPCTL4 . . . . .	228
7.4.169	DOEPINT4 . . . . .	231
7.4.170	DOEPTSIZ4 . . . . .	232
7.4.171	DOEPDMA4 . . . . .	233
7.4.172	DOEPDMAB4 . . . . .	233
7.4.173	DOEPCTL5 . . . . .	234

7.4.174	DOEPINT5 . . . . .	237
7.4.175	DOEPTSIZ5 . . . . .	238
7.4.176	DOEPDMA5 . . . . .	239
7.4.177	DOEPDMAB5 . . . . .	239
7.4.178	DOEPCTL6 . . . . .	240
7.4.179	DOEPINT6 . . . . .	243
7.4.180	DOEPTSIZ6 . . . . .	244
7.4.181	DOEPDMA6 . . . . .	245
7.4.182	DOEPDMAB6 . . . . .	245
7.4.183	DOEPCTL7 . . . . .	246
7.4.184	DOEPINT7 . . . . .	249
7.4.185	DOEPTSIZ7 . . . . .	250
7.4.186	DOEPDMA7 . . . . .	251
7.4.187	DOEPDMAB7 . . . . .	251
7.4.188	DOEPCTL8 . . . . .	252
7.4.189	DOEPINT8 . . . . .	255
7.4.190	DOEPTSIZ8 . . . . .	256
7.4.191	DOEPDMA8 . . . . .	257
7.4.192	DOEPDMAB8 . . . . .	257
7.4.193	DOEPCTL9 . . . . .	258
7.4.194	DOEPINT9 . . . . .	261
7.4.195	DOEPTSIZ9 . . . . .	262
7.4.196	DOEPDMA9 . . . . .	263
7.4.197	DOEPDMAB9 . . . . .	263
7.4.198	DOEPCTL10 . . . . .	264
7.4.199	DOEPINT10 . . . . .	267
7.4.200	DOEPTSIZ10 . . . . .	268
7.4.201	DOEPDMA10 . . . . .	269
7.4.202	DOEPDMAB10 . . . . .	269

7.4.203	DOEPCTL11 . . . . .	270
7.4.204	DOEPINT11 . . . . .	273
7.4.205	DOEPTSIZ11 . . . . .	274
7.4.206	DOEPDMA11 . . . . .	275
7.4.207	DOEPDMAB11 . . . . .	275
7.4.208	DOEPCTL12 . . . . .	276
7.4.209	DOEPINT12 . . . . .	279
7.4.210	DOEPTSIZ12 . . . . .	280
7.4.211	DOEPDMA12 . . . . .	281
7.4.212	DOEPDMAB12 . . . . .	281
7.4.213	DOEPCTL13 . . . . .	282
7.4.214	DOEPINT13 . . . . .	285
7.4.215	DOEPTSIZ13 . . . . .	286
7.4.216	DOEPDMA13 . . . . .	287
7.4.217	DOEPDMAB13 . . . . .	287
7.4.218	DOEPCTL14 . . . . .	288
7.4.219	DOEPINT14 . . . . .	291
7.4.220	DOEPTSIZ14 . . . . .	292
7.4.221	DOEPDMA14 . . . . .	293
7.4.222	DOEPDMAB14 . . . . .	293
7.4.223	DOEPCTL15 . . . . .	294
7.4.224	DOEPINT15 . . . . .	297
7.4.225	DOEPTSIZ15 . . . . .	298
7.4.226	DOEPDMA15 . . . . .	299
7.4.227	DOEPDMAB15 . . . . .	299
<b>8</b>	<b>DDRC</b>	<b>300</b>
8.1	PIN Description . . . . .	300
8.1.1	AXI Interfaces . . . . .	300

8.1.2	APB Interfaces . . . . .	301
8.1.3	DDR Interfaces . . . . .	302
8.1.4	Configuration Done interface . . . . .	302
8.2	DDRC Register Description . . . . .	303
8.2.1	Register Summary . . . . .	303
8.2.2	DDR_PHY_CONFIG . . . . .	304
8.2.3	DDR_ADDR_SIZE . . . . .	305
8.2.4	DDR_TIMING_0 . . . . .	306
8.2.5	DDR_TIMING_1 . . . . .	306
8.2.6	DDR_TIMING_2 . . . . .	306
8.2.7	DDR_LMR_EXT_STD . . . . .	306
8.2.8	DDR_LMR_EXT_3_2 . . . . .	306
8.2.9	DDR_CFG_RDQ_LATENCY . . . . .	307
8.2.10	DDR_CFG_RDQ_DLYVAL . . . . .	307
8.2.11	DDR_CFG_CMD_DLYVAL . . . . .	307
8.2.12	DDR_STS_RDQ_LATENCY . . . . .	308
8.2.13	DDR_STS_RDQ_DLYVAL . . . . .	309
8.2.14	DDR_STS_CMD_DLYVAL . . . . .	309
8.2.15	DDR_STS_VREF_ERRPOS . . . . .	309
8.2.16	DDR_STS_VREF_ERRDATA_L . . . . .	309
8.2.17	DDR_STS_VREF_ERRDATA_H . . . . .	309
8.2.18	DDR_STS_RDQCPT_R_BEAT_0_7 . . . . .	310
8.2.19	DDR_STS_RDQCPT_R_BEAT_8_15 . . . . .	311
8.2.20	DDR_STS_RDQCPT_R_BEAT_16_23 . . . . .	311
8.2.21	DDR_STS_RDQCPT_R_BEAT_24_31 . . . . .	312
8.2.22	DDR_STS_RDQCPT_F_BEAT_0_7 . . . . .	313
8.2.23	DDR_STS_RDQCPT_F_BEAT_8_15 . . . . .	314
8.2.24	DDR_STS_RDQCPT_F_BEAT_16_23 . . . . .	314
8.2.25	DDR_STS_RDQCPT_F_BEAT_24_31 . . . . .	315

8.2.26	DDR_STS_RDQCPT_BEAT_0_7 . . . . .	316
8.2.27	DDR_STS_RDQCPT_BEAT_8_15 . . . . .	317
8.2.28	DDR_STS_RDQCPT_BEAT_16_23 . . . . .	317
8.2.29	DDR_STS_RDQCPT_BEAT_24_31 . . . . .	318
8.2.30	DDR_STS_RDQCPT_BYT0_0_7 . . . . .	319
8.2.31	DDR_STS_RDQCPT_BYT0_8_15 . . . . .	320
8.2.32	DDR_STS_RDQCPT_BYT0_16_23 . . . . .	320
8.2.33	DDR_STS_RDQCPT_BYT0_24_31 . . . . .	321
8.2.34	DDR_STS_RDQCPT_BYT1_0_7 . . . . .	322
8.2.35	DDR_STS_RDQCPT_BYT1_8_15 . . . . .	322
8.2.36	DDR_STS_RDQCPT_BYT1_16_23 . . . . .	323
8.2.37	DDR_STS_RDQCPT_BYT1_24_31 . . . . .	323
8.2.38	DDR_STS_RDQCPT_BYT2_0_7 . . . . .	324
8.2.39	DDR_STS_RDQCPT_BYT2_8_15 . . . . .	324
8.2.40	DDR_STS_RDQCPT_BYT2_16_23 . . . . .	325
8.2.41	DDR_STS_RDQCPT_BYT2_24_31 . . . . .	325
8.2.42	DDR_STS_RDQCPT_BYT3_0_7 . . . . .	326
8.2.43	DDR_STS_RDQCPT_BYT3_8_15 . . . . .	326
8.2.44	DDR_STS_RDQCPT_BYT3_16_23 . . . . .	327
8.2.45	DDR_STS_RDQCPT_BYT3_24_31 . . . . .	327
8.3	Activation GUIDE . . . . .	328
<b>9</b>	<b>HOVER_DMA</b>	<b>329</b>
9.1	Features . . . . .	329
9.2	Functional Description . . . . .	330
9.2.1	Frame buffer handling . . . . .	330
9.2.2	Waveform . . . . .	331
9.3	HOVER DMA Register Description . . . . .	332
9.3.1	Register Map Summary . . . . .	332

9.3.2	INT_ENABLE . . . . .	333
9.3.3	INT_DISABLE . . . . .	333
9.3.4	INT_PEND . . . . .	333
9.3.5	INT_STATUS . . . . .	334
9.3.6	INT_PEND_NUMBER . . . . .	334
9.3.7	CONTROL . . . . .	334
9.3.8	CONFIG . . . . .	334
9.3.9	STATUS . . . . .	335
9.3.10	ADDR . . . . .	335
9.3.11	STRIDE . . . . .	335
9.3.12	WIDTH . . . . .	335
9.3.13	HEIGHT . . . . .	336
9.3.14	HFP . . . . .	336
9.3.15	HBP . . . . .	336
9.3.16	VFP . . . . .	336
9.3.17	VBP . . . . .	337
9.3.18	DONE_ADDR . . . . .	337
<b>10</b>	<b>VIP</b>	<b>338</b>
10.1	Overview . . . . .	339
10.2	Features . . . . .	340
10.3	Interconnection . . . . .	341
10.3.1	External Pad Interconnection . . . . .	341
10.3.2	Clock Generation . . . . .	342
10.4	Video Input Port . . . . .	343
10.4.1	Block Diagram . . . . .	343
10.4.2	Sync Generation . . . . .	343
10.4.3	ITU-R BT.601 . . . . .	344
10.4.4	ITU-R BT.656 . . . . .	345

10.4.5	External Data Valid and Field . . . . .	347
10.4.5.1	External Data Valid . . . . .	347
10.4.5.2	External Field . . . . .	347
10.4.5.3	Data Order . . . . .	347
10.4.5.4	Horizontal & Vergical Counter . . . . .	348
10.4.5.5	Current HSYNC & VSYNC Status . . . . .	348
10.4.5.6	Current Field Status . . . . .	348
10.4.5.7	FIFO Controls . . . . .	348
10.4.5.8	Recommend Setting for Video Input Port . . . . .	348
10.5	Clipper & Decimator . . . . .	349
10.5.1	Clipping & Scale Down . . . . .	349
10.5.2	Interlace Scan mode . . . . .	350
10.5.3	Output Data Format . . . . .	350
10.5.4	Output Address and Stride . . . . .	350
10.5.5	Interrupt Generation . . . . .	350
10.6	VIP Register Map . . . . .	351
10.6.1	Register Map Summary . . . . .	351
10.6.2	VIP0_CONFIG . . . . .	352
10.6.3	VIP0_INTCTRL . . . . .	352
10.6.4	VIP0_FIELD . . . . .	352
10.6.5	VIP0_SYNCMON . . . . .	353
10.6.6	VIP0_VBEGIN . . . . .	353
10.6.7	VIP0_VEND . . . . .	354
10.6.8	VIP0_HBEGIN . . . . .	354
10.6.9	VIP0_HEND . . . . .	354
10.6.10	VIP0_FIFOCTRL . . . . .	355
10.6.11	VIP0_HCOUNTER . . . . .	355
10.6.12	VIP0_VCOUNT . . . . .	355
10.6.13	VIP0 Pad clock invert . . . . .	356

10.6.14	VIP0 INFIFO Clear . . . . .	356
10.7	VIP Prescaler Register Map . . . . .	357
10.7.1	Register Map Summary . . . . .	357
10.7.2	VIP_CDENB . . . . .	358
10.7.3	VIP_ODINT . . . . .	358
10.7.4	VIP_IMGWIDTH . . . . .	358
10.7.5	VIP_IMGHEIGHT . . . . .	359
10.7.6	CLIP_LEFT . . . . .	359
10.7.7	CLIP_RIGHT . . . . .	359
10.7.8	CLIP_TOP . . . . .	359
10.7.9	CLIP_BOTTOM . . . . .	360
10.7.10	DECI_TARGETW . . . . .	360
10.7.11	DECI_TARGETH . . . . .	360
10.7.12	DECI_DELTAW . . . . .	360
10.7.13	DECI_DELTAH . . . . .	361
10.7.14	DECI_CLEARW . . . . .	361
10.7.15	DECI_CLEARH . . . . .	361
10.7.16	DECI_FORMAT . . . . .	361
10.7.17	DECI_LUADDR . . . . .	362
10.7.18	DECI_LUSTRIDE . . . . .	362
10.7.19	DECI_CRADDR . . . . .	362
10.7.20	DECI_CRStride . . . . .	362
10.7.21	DECI_CBADDR . . . . .	363
10.7.22	DECI_CBStride . . . . .	363
10.7.23	CLIP_FORMAT . . . . .	363
10.7.24	CLIP_LUADDR . . . . .	363
10.7.25	CLIP_LUSTRIDE . . . . .	364
10.7.26	CLIP_CRADDR . . . . .	364
10.7.27	CLIP_CRStride . . . . .	364

10.7.28	CLIP_CBADDR . . . . .	364
10.7.29	CLIP_CBStride . . . . .	365
10.7.30	VIP_SCANMODE . . . . .	365
10.7.31	VIP_PORTSEL . . . . .	365
<b>11</b>	<b>Multi-Format Video Codec</b>	<b>366</b>
11.1	Overview . . . . .	366
11.2	Functional Description . . . . .	367
11.2.1	List of Video CODECs . . . . .	367
11.2.2	Supported Video Encoding Tools . . . . .	368
11.2.2.1	H.264/AVC BP/CBP Encoder . . . . .	368
11.2.2.2	MPEG4-SP Encoder . . . . .	368
11.2.2.3	H.263 P0/P3 (Interactive and Streaming Wireless Profile) Encoder . . . . .	369
11.2.3	Supported Video Decoding Tools . . . . .	369
11.2.3.1	H.264/AVC Decoder . . . . .	369
11.2.3.2	VC-1/WMV-9 Decoder . . . . .	369
11.2.3.3	MPEG-4 Decoder . . . . .	369
11.2.3.4	Sorenson Spark Decoder . . . . .	369
11.2.3.5	H.263 V2 (Interactive and Streaming Wireless Profile, Profile 3) Decoder . . . . .	369
11.2.3.6	MPEG-1/MPEG-2 . . . . .	370
11.2.3.7	AVS Decoder . . . . .	370
11.2.3.8	Real Video 10 Decoder . . . . .	370
11.2.3.9	VP8 Decoder . . . . .	370
11.2.3.10	Theora Decoder . . . . .	370
11.2.4	Supported JPEG Tools . . . . .	370
11.2.4.1	MJPEG Baseline Process Encoder and Decoder . . . . .	370
11.2.5	Non-codec related features . . . . .	371
11.2.5.1	Value Added Features . . . . .	371
11.2.5.2	Programmability . . . . .	371

11.2.5.3	Optimal External Memory Accesses . . . . .	371
<b>12 Scaler</b>		<b>372</b>
12.1	Overview . . . . .	372
12.2	Features . . . . .	372
12.2.1	Scaler . . . . .	372
12.3	Block Diagram . . . . .	373
12.4	Functional Description . . . . .	374
12.4.1	Digital Filter Characteristics . . . . .	374
12.4.1.1	Horizontal Filter (5-Tab FIR Filter) Frequency Response and Group Delay . .	374
12.4.1.2	Vertical Filter (3-Tab FIR Filter) Frequency Response and Group Delay . .	375
12.5	Programming Guide . . . . .	376
12.5.1	Configuration . . . . .	376
12.5.2	RUN . . . . .	377
12.6	Register Description . . . . .	378
12.6.1	Register Map Summary . . . . .	378
12.6.2	SCRUNREG . . . . .	380
12.6.3	SCCFGREG . . . . .	380
12.6.4	SCINTREG . . . . .	381
12.6.5	SCSRCADDRREG . . . . .	381
12.6.6	SCSRCADDRREG . . . . .	381
12.6.7	SCSRCSIZEREG . . . . .	382
12.6.8	SCDESTADDR0 . . . . .	382
12.6.9	SCDESTSTREDE0 . . . . .	382
12.6.10	SCDESTADDR1 . . . . .	382
12.6.11	SCDESTSTREDE1 . . . . .	383
12.6.12	SCDESTSIZE . . . . .	383
12.6.13	DELTAZREG . . . . .	383
12.6.14	DELTAZREG . . . . .	383

12.6.15	HVSOFTRREG . . . . .	384
12.6.16	CMDBUFADDR . . . . .	384
12.6.17	CMDBUFCON . . . . .	384
12.6.18	YVFILTER[N]_00_03 . . . . .	385
12.6.19	YVFILTER[N]_04_07 . . . . .	385
12.6.20	YVFILTER[N]_08_11 . . . . .	385
12.6.21	YVFILTER[N]_12_15 . . . . .	386
12.6.22	YVFILTER[N]_16_19 . . . . .	386
12.6.23	YVFILTER[N]_20_23 . . . . .	386
12.6.24	YVFILTER[N]_24_27 . . . . .	387
12.6.25	YVFILTER[N]_28_31 . . . . .	387
12.6.26	YHFILTER[N]_00_01 . . . . .	388
12.6.27	YHFILTER[N]_02_03 . . . . .	388
12.6.28	YHFILTER[N]_04_05 . . . . .	388
12.6.29	YHFILTER[N]_06_07 . . . . .	389
12.6.30	YHFILTER[N]_08_09 . . . . .	389
12.6.31	YHFILTER[N]_10_11 . . . . .	389
12.6.32	YHFILTER[N]_12_13 . . . . .	390
12.6.33	YHFILTER[N]_14_15 . . . . .	390
12.6.34	YHFILTER[N]_16_17 . . . . .	390
12.6.35	YHFILTER[N]_18_19 . . . . .	391
12.6.36	YHFILTER[N]_20_21 . . . . .	391
12.6.37	YHFILTER[N]_22_23 . . . . .	391
12.6.38	YHFILTER[N]_24_25 . . . . .	392
12.6.39	YHFILTER[N]_26_27 . . . . .	392
12.6.40	YHFILTER[N]_28_29 . . . . .	392
12.6.41	YHFILTER[N]_30_31 . . . . .	393
<b>13</b>	<b>Crypto Engine</b>	<b>394</b>

13.1	Overview	394
13.2	Features	394
13.2.1	Crypto Engine	394
13.3	Block Diagram	395
13.4	Functional Description	397
13.4.1	Polling Mode	397
13.4.2	Mode	397
13.5	Register Description	398
13.5.1	Register Map Summary	398
13.5.2	CRT_CTRL0	400
13.5.3	AES_CTRL0	401
13.5.4	AES_iv0	402
13.5.5	AES_iv1	402
13.5.6	AES_iv2	402
13.5.7	AES_iv3	402
13.5.8	AES_key0	403
13.5.9	AES_key1	403
13.5.10	AES_key2	403
13.5.11	AES_key3	403
13.5.12	AES_key4	404
13.5.13	AES_key5	404
13.5.14	AES_key6	404
13.5.15	AES_key7	404
13.5.16	AES_TEXTINO	405
13.5.17	AES_TEXTIN1	405
13.5.18	AES_TEXTIN2	405
13.5.19	AES_TEXTIN3	405
13.5.20	AES_TEXTOUT0	406
13.5.21	AES_TEXTOUT1	406

13.5.22	AES_TEXTOUT2	406
13.5.23	AES_TEXTOUT3	406
13.5.24	DES_CTRL0	407
13.5.25	DES_iv0	408
13.5.26	DES_iv1	408
13.5.27	DES_KEY0_0	408
13.5.28	DES_KEY0_1	408
13.5.29	DES_KEY1_0	409
13.5.30	DES_KEY1_1	409
13.5.31	DES_KEY2_0	409
13.5.32	DES_KEY2_1	409
13.5.33	DES_TEXTIN0	410
13.5.34	DES_TEXTIN1	410
13.5.35	DES_TEXTOUT0	410
13.5.36	DES_TEXTOUT1	410
13.5.37	BDMAR	411
13.5.38	BDMAW	411
13.5.39	HDMAR	411
13.5.40	HASH_CTRL0	412
13.5.41	HASH_iv0	412
13.5.42	HASH_iv1	412
13.5.43	HASH_iv2	413
13.5.44	HASH_iv3	413
13.5.45	HASH_iv4	413
13.5.46	HASH_TEXTOUT0	414
13.5.47	HASH_TEXTOUT1	414
13.5.48	HASH_TEXTOUT2	414
13.5.49	HASH_TEXTOUT3	414
13.5.50	HASH_TEXTOUT4	415

13.5.51	HASH_TEXTIN . . . . .	415
13.5.52	HASH_MSG_SIZE . . . . .	415
13.5.53	HASH_MSG_SIZE . . . . .	415
<b>14 ADC</b>		<b>416</b>
14.1	Overview . . . . .	416
14.2	Functional Description . . . . .	416
14.3	Features . . . . .	417
14.4	Connection Circuit . . . . .	418
14.5	Timing Characteristics . . . . .	419
14.6	Timing Characteristics (Case I) . . . . .	421
14.7	Timing Characteristics (Case II) . . . . .	423
14.8	Pin Descriptions . . . . .	425
14.9	ADC Register Description . . . . .	426
14.9.1	Register Map Summary . . . . .	426
14.9.2	ADCCON . . . . .	427
14.9.3	ADCDAT . . . . .	427
14.9.4	ADCINTENB . . . . .	428
14.9.5	ADCINTCLR . . . . .	428
14.9.6	PRESCALERCON . . . . .	428
14.9.7	ADCEN . . . . .	429
<b>15 I2C</b>		<b>430</b>
15.1	Product Overview . . . . .	430
15.1.1	DesignWare System Overview . . . . .	431
15.1.2	General Product Description . . . . .	433
15.1.2.1	DW_apb_i2c Block Diagram . . . . .	433
15.1.3	Features . . . . .	434
15.1.3.1	I2C Features . . . . .	434
15.1.3.2	DesignWare APB Slave Interface . . . . .	435

15.1.4	Standards Compliance . . . . .	435
15.2	I2C Functional Description . . . . .	436
15.2.1	Overview . . . . .	436
15.2.2	I2C Terminology . . . . .	438
15.2.2.1	I2C Bus Terms . . . . .	438
15.2.2.2	Bus Transfer Terms . . . . .	439
15.2.3	I2C Behavior . . . . .	439
15.2.3.1	START and STOP Generation . . . . .	440
15.2.3.2	Combined Formats . . . . .	440
15.2.4	I2C Protocols . . . . .	441
15.2.4.1	START and STOP Conditions . . . . .	441
15.2.4.2	Addressing Slave Protocol . . . . .	441
15.2.4.2.1	7-bit Address Format . . . . .	441
15.2.4.2.2	10-bit Address Format . . . . .	442
15.2.4.3	Transmitting and Receiving Protocol . . . . .	443
15.2.4.3.1	Master-Transmitter and Slave-Receiver . . . . .	443
15.2.4.3.2	Master-Receiver and Slave-Transmitter . . . . .	444
15.2.4.4	START BYTE Transfer Protocol . . . . .	445
15.2.5	Tx FIFO Management and START, STOP and RESTART Generation . . . . .	446
15.2.5.1	Tx FIFO Management When IC_EMPTYFIFO_HOLD_MASTER_EN = 0 . . . . .	446
15.2.6	Tx FIFO Management When IC_EMPTYFIFO_HOLD_MASTER_EN = 1 . . . . .	447
15.2.7	Multiple Master Arbitration . . . . .	452
15.2.8	Clock Synchronization . . . . .	453
15.2.9	Operation Modes . . . . .	454
15.2.9.1	Slave Mode Operation . . . . .	454
15.2.9.1.1	Initial Configuration . . . . .	454
15.2.9.1.2	Slave-Transmitter Operation for a Single Byte . . . . .	455
15.2.9.1.3	Slave-Receiver Operation for a Single Byte . . . . .	456
15.2.9.1.4	Slave-Transfer Operation For Bulk Transfers . . . . .	456

15.2.9.2	Master Mode Operation . . . . .	457
15.2.9.2.1	Initial Configuration . . . . .	457
15.2.9.2.1.1	I2C_DYNAMIC_TAR_UPDATE = 0 . . . . .	457
15.2.9.2.1.2	I2C_DYNAMIC_TAR_UPDATE = 1 . . . . .	458
15.2.9.2.2	Dynamic IC_TAR or IC_10BITADDR_MASTER Update . . . . .	458
15.2.9.2.3	Master Transmit and Master Receive . . . . .	459
15.2.9.3	Disabling DW_apb_i2c . . . . .	459
15.2.9.3.1	Procedure . . . . .	460
15.2.9.4	Aborting I2C Transfers . . . . .	460
15.2.9.4.1	Procedure . . . . .	460
15.2.9.5	Spike Suppression . . . . .	461
15.2.10	Fast Mode Plus Operation . . . . .	462
15.2.11	Bus Clear Feature . . . . .	463
15.2.11.1	SDA Line Stuck at LOW Recovery . . . . .	463
15.2.11.2	SCL Line is Stuck at LOW . . . . .	464
15.2.12	Device ID . . . . .	464
15.2.13	Ultra-Fast Speed Mode . . . . .	465
15.2.14	SMBus/PMBus . . . . .	466
15.2.14.1	tTimeout,MIN Parameter . . . . .	466
15.2.14.2	Master Device Clock Extension . . . . .	466
15.2.14.3	Slave Device Clock Extension . . . . .	467
15.2.14.4	SMBDAT Low Timeout . . . . .	467
15.2.14.5	Bus Protocols . . . . .	467
15.2.14.6	SMBUS Address Resolution Protocol . . . . .	469
15.2.14.6.1	Procedure to Perform ARP in Master Mode . . . . .	473
15.2.14.6.2	Procedure to Perform ARP in Slave Mode . . . . .	474
15.2.14.7	SMBUS Additional Slave Address . . . . .	476
15.2.14.8	SMBUS Optional Signals . . . . .	476
15.2.14.8.1	SMBus Suspend Signal . . . . .	477

15.2.14.8.2SMBus Alert Signal . . . . .	477
15.2.15 IC_CLK Frequency Configuration . . . . .	477
15.2.15.1 Minimum High and Low Counts in SS, FS, FM+ and HS Modes With IC_CLK_FREQ_OPTIMIZATION = 0 . . . . .	479
15.2.15.2 Minimum High and Low Counts in SS, FS, FM+ and HS Modes With IC_CLK_FREQ_OPTIMIZATION = 1 . . . . .	480
15.2.15.3 Minimum High and Low counts in Ultra-Fast mode (IC_ULTRA_FAST_MODE = 1) . . . . .	480
15.2.15.4 Minimum IC_CLK Frequency . . . . .	481
15.2.15.4.1 Standard Mode (SM), Fast Mode (FM), and Fast Mode Plus (FM+) with IC_CLK_FREQ_OPTIMIZATION = 0 . . . . .	481
15.2.15.4.2 High-Speed (HS) Mode With IC_CLK_FREQ_OPTIMIZATION = 0 . . . . .	481
15.2.15.4.3 SM, FM, FM+ and HS Modes With IC_CLK_FREQ_OPTIMIZATION = 1 . . . . .	482
15.2.15.4.3.1 Master Mode . . . . .	482
15.2.15.4.3.2 Slave Mode . . . . .	484
15.2.15.4.4 ULTRA-FAST Mode . . . . .	484
15.2.15.4.4.1 Master mode . . . . .	484
15.2.15.4.4.2 Slave mode . . . . .	485
15.2.15.4.5 Calculating High and Low Counts with IC_CLK_FREQ_OPTIMIZATION = 0 . . . . .	485
15.2.15.4.6 Calculating High and Low counts with IC_CLK_FREQ_OPTIMIZATION = 1 . . . . .	486
15.2.16 SDA Hold Time . . . . .	487
15.2.16.1 SDA Hold Timings in Receiver . . . . .	487
15.2.16.2 SDA Hold Timings in Transmitter . . . . .	489
15.2.17 DMA Controller Interface . . . . .	490
15.2.17.1 Enabling the DMA Controller Interface . . . . .	490
15.2.17.2 Overview of Operation . . . . .	491
15.2.17.3 Transmit Watermark Level and Transmit FIFO Underflow . . . . .	492
15.2.17.4 Choosing the Transmit Watermark Level . . . . .	492
15.2.17.4.1 Case 1: IC_DMA_TDLR = 2 . . . . .	492
15.2.17.4.2 Case 2: IC_DMA_TDLR = 6 . . . . .	493
15.2.17.5 Selecting DEST_MSIZE and Transmit FIFO Overflow . . . . .	494

15.2.17.6	Receive Watermark Level and Receive FIFO Overflow . . . . .	494
15.2.17.7	Choosing the Receive Watermark level . . . . .	494
15.2.17.8	Selecting SRC_MSIZE and Receive FIFO Underflow . . . . .	494
15.2.17.9	Handshaking Interface Operation . . . . .	495
15.2.17.9.1	dma_tx_req, dma_rx_req . . . . .	495
15.2.17.9.2	dma_tx_single, dma_rx_single . . . . .	497
15.2.18	APB Interface . . . . .	498
15.2.19	I/O Connections . . . . .	499
15.2.20	DW_apb_i2c Registers . . . . .	499
15.2.20.1	Registers and Field Descriptions . . . . .	499
15.2.20.2	Operation of Interrupt Registers . . . . .	500
15.3	Parameter Descriptions . . . . .	501
15.3.1	Top Level Parameters . . . . .	501
15.3.2	I2C Version 3.0 Features Parameters . . . . .	508
15.3.3	SMBus Features Parameters . . . . .	508
15.3.4	I2C Version 6.0 Features Parameters . . . . .	509
15.4	Signal Descriptions . . . . .	510
15.4.1	Interrupts Signals . . . . .	511
15.4.2	I2C Interface (Master/Slave) Signals . . . . .	518
15.4.3	APB Slave Interface Signals . . . . .	521
15.4.4	DMA Interface Signals . . . . .	523
15.4.5	SMBus Interface Signals . . . . .	525
15.4.6	I2C Debug Signals . . . . .	526
15.5	I2C Register Description . . . . .	528
15.5.1	Register Map Summary . . . . .	528
15.5.2	I2C Control Register . . . . .	531
15.5.3	I2C Target Address Register . . . . .	534
15.5.4	I2C Slave Address Register . . . . .	535
15.5.5	I2C High Speed Master Mode Code Address Register . . . . .	536

15.5.6	I2C Rx/Tx Data Buffer and Command Register . . . . .	537
15.5.7	Standard Speed I2C Clock SCL High Count Register . . . . .	538
15.5.8	Ultra-Fast Speed I2C Clock SCL High Count Register . . . . .	539
15.5.9	Standard Speed I2C Clock SCL Low Count Register . . . . .	540
15.5.10	Ultra-Fast Speed I2C Clock SCL Low Count Register . . . . .	541
15.5.11	Fast Mode or Fast Mode Plus I2C Clock SCL High Count Register . . . . .	541
15.5.12	Ultra-Fast Speed mode TBuf Idle Count Register . . . . .	542
15.5.13	Fast Mode or Fast Mode Plus I2C Clock SCL Low Count Register . . . . .	543
15.5.14	High Speed I2C Clock SCL High Count Register . . . . .	544
15.5.15	High Speed I2C Clock SCL Low Count Register . . . . .	544
15.5.16	I2C Interrupt Status Register . . . . .	545
15.5.17	I2C Interrupt Mask Register . . . . .	547
15.5.18	I2C Raw Interrupt Status Register . . . . .	549
15.5.19	I2C Receive FIFO Threshold Register . . . . .	553
15.5.20	I2C Transmit FIFO Threshold Register . . . . .	554
15.5.21	Clear Combined and Individual Interrupt Register . . . . .	554
15.5.22	Clear RX_UNDER Interrupt Register . . . . .	555
15.5.23	Clear RX_OVER Interrupt Register . . . . .	555
15.5.24	Clear TX_OVER Interrupt Register . . . . .	556
15.5.25	Clear RD_REQ Interrupt Register . . . . .	557
15.5.26	Clear TX_ABRT Interrupt Register . . . . .	557
15.5.27	Clear RX_DONE Interrupt Register . . . . .	558
15.5.28	Clear ACTIVITY Interrupt Register . . . . .	558
15.5.29	Clear STOP_DET Interrupt Register . . . . .	559
15.5.30	Clear START_DET Interrupt Register . . . . .	560
15.5.31	Clear GEN_CALL Interrupt Register . . . . .	560
15.5.32	I2C ENABLE Register . . . . .	561
15.5.33	I2C STATUS Register . . . . .	563
15.5.34	I2C Transmit FIFO Level Register . . . . .	566

15.5.35	I2C Receive FIFO Level Register . . . . .	567
15.5.36	I2C SDA Hold Time Length Register . . . . .	567
15.5.37	I2C SDA Hold Time Length Register . . . . .	568
15.5.38	Generate Slave Data NACK Register . . . . .	572
15.5.39	DMA Control Register . . . . .	573
15.5.40	DMA Transmit Data Level Register . . . . .	574
15.5.41	DMA Transmit Data Level Register . . . . .	574
15.5.42	I2C SDA Setup Register . . . . .	575
15.5.43	I2C ACK General Call Register . . . . .	576
15.5.44	I2C Enable Status Register . . . . .	576
15.5.45	I2C SS, FS or FM+ spike suppression limit . . . . .	578
15.5.46	I2C Ultra-Fast mode spike suppression limit . . . . .	578
15.5.47	I2C HS spike suppression limit register . . . . .	579
15.5.48	Clear RESTART_DET Interrupt Register . . . . .	580
15.5.49	I2C SCL Stuck at Low Timeout register . . . . .	580
15.5.50	I2C SDA Stuck at Low Timeout register . . . . .	581
15.5.51	Clear SCL Stuck at Low Detect interrupt Register . . . . .	581
15.5.52	I2C Device-ID Register . . . . .	582
15.5.53	SMBus Slave Clock Extend Timeout register . . . . .	583
15.5.54	SMBus Master Clock Extend Timeout register . . . . .	583
15.5.55	SMBus Master THigh MAX Bus-idle count Register . . . . .	584
15.5.56	SMBus Interrupt Status Register . . . . .	584
15.5.57	SMBus Interrupt Mask Register . . . . .	586
15.5.58	SMBus Raw Interrupt Status Register . . . . .	588
15.5.59	Clear SMBus Interrupt Register . . . . .	590
15.5.60	I2C Optional Slave Address Register . . . . .	591
15.5.61	SMBUS ARP UDID LSB Register . . . . .	592
15.5.62	Component Parameter Register 1 . . . . .	592
15.5.63	I2C Component Version Register . . . . .	594

15.5.64	I2C Component Type Register . . . . .	594
15.6	Programming the DW_apb_i2c . . . . .	595
15.6.1	Software Registers . . . . .	595
15.6.2	Software Drivers . . . . .	595
15.6.3	Programming Example . . . . .	596
15.6.4	Programming Flow for SCL and SDA Bus Recovery . . . . .	602
15.6.5	Programming Flow for Reading the Device ID . . . . .	603
15.6.6	Programming Flow for SMBUS Timeout in Master Mode . . . . .	604
15.6.7	Programming Flow for SMBUS Timeout in Slave Mode . . . . .	605
15.6.8	ARP Master Programming Flow . . . . .	606
15.6.9	ARP Slave Programming Flow . . . . .	607
15.6.10	SMBus SUSPEND Programming Flow in Host Mode . . . . .	609
15.6.11	SMBus SUSPEND Programming Flow in Device Mode . . . . .	610
15.6.12	SMBus ALERT Programming Flow in Host Mode . . . . .	611
15.6.13	SMBus ALERT Programming Flow in Device Mode . . . . .	612
15.6.14	Programming Flow Of DW_apb_i2c in Ultra-Fast Mode . . . . .	613
15.6.14.1	DW_apb_i2c Master Mode . . . . .	613
15.6.14.2	DW_apb_i2c Slave Mode . . . . .	614
<b>16</b>	<b>SDMMC</b> . . . . .	<b>615</b>
16.1	Overview . . . . .	615
16.2	Features . . . . .	615
16.3	Block Diagram . . . . .	617
16.4	Clock Logic . . . . .	619
16.5	Programmer's Guide . . . . .	620
16.5.1	General Restrictions . . . . .	620
16.5.2	Clock Programming . . . . .	620
16.5.2.1	Enable and Disable Clock . . . . .	620
16.5.2.2	CIU Update . . . . .	621

16.5.3	Sending a Command . . . . .	621
16.5.4	Command Response . . . . .	621
16.5.5	IDMAC Descriptor . . . . .	622
16.5.6	Register Map Summary . . . . .	625
16.5.7	CTRL . . . . .	627
16.5.8	PWREN . . . . .	628
16.5.9	CLKDIV . . . . .	629
16.5.10	CLKSRC . . . . .	629
16.5.11	CLKENA . . . . .	630
16.5.12	TMOUT . . . . .	630
16.5.13	CTYPE . . . . .	630
16.5.14	BLKSIZ . . . . .	631
16.5.15	BYTCNT . . . . .	631
16.5.16	INTMASK . . . . .	632
16.5.17	CMDARG . . . . .	632
16.5.18	CMD . . . . .	633
16.5.19	RESP0 . . . . .	635
16.5.20	RESP1 . . . . .	635
16.5.21	RESP2 . . . . .	635
16.5.22	RESP3 . . . . .	635
16.5.23	MINTSTS . . . . .	636
16.5.24	RINTSTS . . . . .	637
16.5.25	STATUS . . . . .	638
16.5.26	FIFOTH . . . . .	639
16.5.27	CDETECT . . . . .	640
16.5.28	WRTPRT . . . . .	640
16.5.29	GPIO . . . . .	641
16.5.30	TCBCNT . . . . .	641
16.5.31	TBBCNT . . . . .	641

16.5.32	DEBNCE	642
16.5.33	USRID	642
16.5.34	VERID	642
16.5.35	HCON	643
16.5.36	UHS_REG	644
16.5.37	RST_n	644
16.5.38	BMOD	645
16.5.39	PLDMND	645
16.5.40	DBADDRL	646
16.5.41	DBADDRL	646
16.5.42	IDSTS	647
16.5.43	IDINTEN	648
16.5.44	DSCADDRL	648
16.5.45	DSCADDRL	649
16.5.46	BUFADDRL	649
16.5.47	BUFADDRL	649
16.5.48	CardThrCtl	650
16.5.49	Back_end_power	650
16.5.50	EMMC_DDR_REG	651
16.5.51	ENABLE_SHIFT	651
16.5.52	TIEOFF_MODE	651
16.5.53	TIEOFF_SRAM	652
16.5.54	TIEOFF_DRV_PHASE	652
16.5.55	TIEOFF_SMP_PHASE	653
16.5.56	TIEOFF_DS_DELAY	653

<b>17 DMA</b>	<b>654</b>	
17.1	Overview	654
17.2	Features	654

17.3	Peripheral DMA Request ID . . . . .	654
17.3.1	DMAC0 . . . . .	655
17.3.2	DMAC1 . . . . .	655
17.4	Instruction . . . . .	656
17.4.1	Key Instruction . . . . .	660
17.4.1.1	DMAMOV . . . . .	660
17.4.1.2	DMAMOV, DMALDP . . . . .	662
17.4.1.3	DMAST, DMASTP . . . . .	662
17.4.1.4	DMASTZ . . . . .	662
17.4.1.5	DMALP, DMALPEND . . . . .	662
17.4.1.6	DMAWFP . . . . .	662
17.4.1.7	DMAFLUSHP . . . . .	662
17.4.1.8	DMAEND . . . . .	662
17.4.2	USAGE Model . . . . .	663
17.4.2.1	Using debug SFRs . . . . .	663
17.4.2.2	Security Scheme . . . . .	664
17.4.2.3	Interrupts . . . . .	664
17.4.2.4	Summary . . . . .	664
17.5	Register Description . . . . .	666
17.5.1	Register Map Summary . . . . .	666
17.5.2	DSR . . . . .	670
17.5.3	DPC . . . . .	670
17.5.4	INTEN . . . . .	671
17.5.5	INT_EVENT_RIS . . . . .	671
17.5.6	INTMIS . . . . .	671
17.5.7	INTCLR . . . . .	672
17.5.8	FSRD . . . . .	672
17.5.9	FSRC . . . . .	672
17.5.10	FTRD . . . . .	673

17.5.11	FTR0	673
17.5.12	FTR2	675
17.5.13	FTR2	675
17.5.14	FTR3	675
17.5.15	FTR4	676
17.5.16	FTR5	676
17.5.17	FTR6	676
17.5.18	FTR7	676
17.5.19	CSR0	677
17.5.20	CPC0	677
17.5.21	CSR1	678
17.5.22	CPC1	678
17.5.23	CSR2	678
17.5.24	CPC2	678
17.5.25	CSR3	679
17.5.26	CPC3	679
17.5.27	CSR4	679
17.5.28	CPC4	679
17.5.29	CSR5	680
17.5.30	CPC5	680
17.5.31	CSR6	680
17.5.32	CPC6	680
17.5.33	CSR7	681
17.5.34	CPC7	681
17.5.35	SAR0	681
17.5.36	DAR0	681
17.5.37	CCR0	682
17.5.38	LC0_0	683
17.5.39	LC1_0	684

17.5.40	SAR2	684
17.5.41	DAR2	684
17.5.42	CCR2	684
17.5.43	LC0_2	685
17.5.44	LC1_2	685
17.5.45	SAR2	685
17.5.46	DAR2	685
17.5.47	CCR2	686
17.5.48	LC0_2	686
17.5.49	LC1_2	686
17.5.50	SAR3	686
17.5.51	DAR3	687
17.5.52	CCR3	687
17.5.53	LC0_3	687
17.5.54	LC1_3	687
17.5.55	SAR4	688
17.5.56	DAR4	688
17.5.57	CCR4	688
17.5.58	LC0_4	688
17.5.59	LC1_4	689
17.5.60	SAR5	689
17.5.61	DAR5	689
17.5.62	CCR5	689
17.5.63	LC0_5	690
17.5.64	LC1_5	690
17.5.65	SAR6	690
17.5.66	DAR6	690
17.5.67	CCR6	691
17.5.68	LC0_6	691

17.5.69	LC1_6	691
17.5.70	SAR7	691
17.5.71	DAR7	692
17.5.72	CCR7	692
17.5.73	LC0_7	692
17.5.74	LC1_7	692
17.5.75	DBGSTATUS	693
17.5.76	DBGCMD	693
17.5.77	DBGINST0	693
17.5.78	DBGINST1	694
17.5.79	CR0	694
17.5.80	CR1	695
17.5.81	CR2	695
17.5.82	CR3	695
17.5.83	CR4	696
17.5.84	CRD	696
17.5.85	WD	697
17.5.86	periph_id_0	697
17.5.87	periph_id_2	697
17.5.88	periph_id_2	698
17.5.89	periph_id_3	698
17.5.90	pcell_id_0	698
17.5.91	pcell_id_2	699
17.5.92	pcell_id_2	699
17.5.93	pcell_id_3	699

**18 GPIO****700**

18.1	Overview	700
18.1.1	Block Diagram	700

18.2	Functional Description	702
18.2.1	Input Operation	702
18.2.2	Output Operation	703
18.2.3	Alternate Function Operation	703
18.2.4	Secure / Non-secure Operation	703
18.3	GPIO Register Description	704
18.3.1	Register Map Summary	704
18.3.2	GPIOxOUT	705
18.3.3	GPIOxOUTENB	705
18.3.4	GPIOxDETMODE0	706
18.3.5	GPIOxDETMODE1	708
18.3.6	GPIOxINTENB	711
18.3.7	GPIOxDET	711
18.3.8	GPIOxPAD	712
18.3.9	GPIOxALTFN0	712
18.3.10	GPIOxALTFN1	713
18.3.11	GPIOxDETMODEEX	714
18.3.12	GPIOxDETENB	719
18.3.13	GPIOx_SLEW	720
18.3.14	GPIOx_SLEW_DISABLE_DEFAULT	720
18.3.15	GPIOx_DRV1	721
18.3.16	GPIOx_DRV1_DISABLE_DEFAULT	721
18.3.17	GPIOx_DRV0	722
18.3.18	GPIOx_DRV0_DISABLE_DEFAULT	722
18.3.19	GPIOx_PULLSEL	723
18.3.20	GPIOx_PULLSEL_DISABLE_DEFAULT	723
18.3.21	GPIOx_PULLENB	724
18.3.22	GPIOx_PULLENB_DISABLE_DEFAULT	724
18.3.23	GPIOx SECURE MARKING REGISTER	725

18.3.24	GPIOx_INPUTENB . . . . .	725
18.3.25	GPIOx_INPUTENB_DISABLE_DEFAULT . . . . .	726
<b>19</b>	<b>SPI/QSPI</b>	<b>727</b>
19.1	Overview . . . . .	727
19.1.1	Features . . . . .	727
19.1.2	Block Diagram . . . . .	728
19.2	Functional Description . . . . .	729
19.2.1	Reset Configuration . . . . .	729
19.2.2	Clock Configuration . . . . .	729
19.2.3	Clock Ratios . . . . .	730
19.2.3.1	Frequency Ratio Summary . . . . .	732
19.2.4	Transfer Modes . . . . .	733
19.2.4.1	Transmit and Receive . . . . .	733
19.2.4.2	Transmit Only . . . . .	733
19.2.4.3	Receive Only . . . . .	733
19.2.4.4	EEPROM Read . . . . .	733
19.2.5	Operation Modes . . . . .	734
19.2.5.1	Serial Master Mode . . . . .	734
19.2.5.1.1	Data Transfers . . . . .	734
19.2.5.1.2	Master SPI and SSP Serial Transfers . . . . .	734
19.2.5.1.3	Master Microwire Serial Transfers . . . . .	736
19.2.5.2	Serial-Slave Mode . . . . .	737
19.2.5.2.1	Slave SPI and SSP Serial Transfers . . . . .	739
19.2.5.2.2	Slave Microwire Serial Transfers . . . . .	741
19.2.6	Partner Connection Interfaces . . . . .	742
19.2.6.1	Motorola Serial Peripheral Interface (SPI) . . . . .	742
19.2.6.2	Texas Instruments Synchronous Serial Protocol (SSP) . . . . .	747
19.2.6.3	Enhanced SPI Modes . . . . .	748

19.2.6.3.1	Write Operation in Enhanced SPI Modes . . . . .	748
19.2.6.3.2	Read Operation in Enhanced SPI Modes . . . . .	750
19.2.6.3.3	Advanced I/O Mapping for Enhanced SPI Modes . . . . .	753
19.2.6.4	Dual Data-Rate (DDR) Support in SPI Operation . . . . .	754
19.2.6.4.1	Transmitting Data in DDR Mode . . . . .	755
19.2.6.5	XIP Mode Support in SPI Mode . . . . .	757
19.2.6.5.1	Read Operation in XIP Mode . . . . .	757
19.2.7	DMA Controller Interface . . . . .	758
19.2.7.1	Overview of Operation . . . . .	760
19.2.7.2	Transmit Watermark Level and Transmit FIFO Underflow . . . . .	760
19.2.7.3	Choosing the Transmit Watermark Level . . . . .	761
19.2.7.3.1	Case 1: DMATDLR = 2 . . . . .	761
19.2.7.3.2	Case 2: DMATDLR = 6 . . . . .	762
19.2.7.4	Selecting DEST_MSIZE and Transmit FIFO Overflow . . . . .	763
19.2.7.5	Receive Watermark Level and Receive FIFO Overflow . . . . .	763
19.2.7.6	Choosing the Receive Watermark level . . . . .	763
19.2.7.7	Selecting SRC_MSIZE and Receive FIFO Underflow . . . . .	764
19.2.7.8	Handshaking Interface Operation . . . . .	764
19.2.7.8.1	dma_tx_req, dma_rx_req . . . . .	764
19.2.7.8.2	dma_tx_single, dma_rx_single . . . . .	765
19.3	Register Description . . . . .	767
19.3.1	Register Map Summary . . . . .	767
19.3.2	CTRLR0 . . . . .	768
19.3.3	CTRLR1 . . . . .	771
19.3.4	SSIENR . . . . .	772
19.3.5	MWCR . . . . .	772
19.3.6	SER . . . . .	773
19.3.7	BAUDR . . . . .	774
19.3.8	TXFTLR . . . . .	775

19.3.9	RXFTLR . . . . .	775
19.3.10	TXFLR . . . . .	776
19.3.11	RXFLR . . . . .	776
19.3.12	SR . . . . .	777
19.3.13	IMR . . . . .	778
19.3.14	ISR . . . . .	779
19.3.15	RISR . . . . .	780
19.3.16	TXOICR . . . . .	780
19.3.17	RXOICR . . . . .	781
19.3.18	RXUICR . . . . .	781
19.3.19	MSTICR . . . . .	782
19.3.20	ICR . . . . .	782
19.3.21	DMACR . . . . .	783
19.3.22	DMATDLR . . . . .	783
19.3.23	DMARDLR . . . . .	784
19.3.24	IDR . . . . .	784
19.3.25	SSI_VERSION_ID . . . . .	785
19.3.26	DRx . . . . .	785
19.3.27	RX_SAMPLE_DLY . . . . .	786
19.3.28	SPI_CTRLR0 . . . . .	786
19.3.29	TXD_DRIVE_EDGE . . . . .	787
19.3.30	RSVD . . . . .	788
<b>20</b>	<b>UART/USART</b> . . . . .	<b>789</b>
20.1	Features . . . . .	789
20.2	Block Diagram . . . . .	791
20.3	Functional Description . . . . .	793
20.3.1	UART (RS232) Serial Protocol . . . . .	793
20.3.2	9-bit Data Transfer . . . . .	794

20.3.2.1	Transmit Mode . . . . .	795
20.3.2.1.1	Transmit Mode 0 . . . . .	795
20.3.2.1.2	Transmit Mode 1 . . . . .	796
20.3.2.2	Receive Mode . . . . .	796
20.3.2.2.1	Hardware Address Match Receive Mode . . . . .	797
20.3.2.2.2	Software Address Match Receive Mode . . . . .	798
20.3.3	RS485 Serial Protocol . . . . .	798
20.3.3.1	DE Assertion and De-assertion Timing . . . . .	798
20.3.3.2	RS485 Modes . . . . .	799
20.3.3.2.1	Full Duplex Mode . . . . .	799
20.3.3.2.2	Software-Controlled Half Duplex Mode . . . . .	800
20.3.3.2.2.1	RE to DE Turnaround Time . . . . .	800
20.3.3.2.2.3	DE to RE Turnaround Time . . . . .	801
20.3.3.2.4	Hardware-Controlled Half Duplex Mode . . . . .	801
20.3.3.3	Sample Scenarios . . . . .	802
20.3.3.3.1	Normal Scenario of Transmission . . . . .	802
20.3.3.3.2	Scenario When Receive is in Progress While TX FIFO is Being Filled . . . . .	803
20.3.3.3.3	TX FIFO Filled Before Enabling DE_EN and RE_EN Registers . . . . .	804
20.3.4	Fractional Baud Rate Support . . . . .	805
20.3.4.1	Fractional Division Used to Generate Baud Clock . . . . .	806
20.3.4.2	Calculating the Fractional Value Error . . . . .	807
20.3.4.2.1	Timing Waveforms . . . . .	808
20.3.5	IrDA 1.0 SIR Protocol . . . . .	808
20.3.6	FIFO Support . . . . .	810
20.3.7	Clock Support . . . . .	812
20.3.8	Back-to-Back Character Stream Transmission . . . . .	814
20.3.8.1	Dual Clock Mode . . . . .	814
20.3.8.2	Single Clock Mode . . . . .	816
20.3.9	Interrupts . . . . .	816

20.3.10	Auto Flow Control . . . . .	818
20.3.11	Programmable THRE Interrupt . . . . .	821
20.3.12	Clock Gate Enable . . . . .	824
20.3.13	DMA Support . . . . .	826
20.3.13.1	DMA Modes . . . . .	826
20.3.13.1.1	DMA Mode 0 . . . . .	827
20.3.13.1.2	DMA Mode 1 . . . . .	827
20.3.13.1.3	Additional DMA Interface . . . . .	827
20.3.13.1.4	Example DMA Flow . . . . .	828
20.3.13.2	Transmit Watermark Level and Transmit FIFO Underflow . . . . .	828
20.3.13.3	Choosing Transmit Watermark Level . . . . .	830
20.3.13.3.1	Case 1: FCR[5:4] = 01 — decodes to 2 . . . . .	830
20.3.13.3.2	Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8) . . . . .	831
20.3.13.4	Selecting DEST_MSIZE and Transmit FIFO Overflow . . . . .	832
20.3.13.5	Receive Watermark Level and Receive FIFO Overflow . . . . .	832
20.3.13.6	Choosing the Receive Watermark Level . . . . .	832
20.3.13.7	Selecting SRC_MSIZE and Receive FIFO Underflow . . . . .	832
20.3.13.8	Handshaking Interface Operation . . . . .	833
20.3.13.9	Potential Deadlock Conditions in UART/DW_ahb_dmac Systems . . . . .	836
20.3.13.9.1	Deadlock When DMA Burst Transaction Length Smaller Than Rx FIFO Threshold . . . . .	837
20.3.13.9.2	Deadlock When DMA Burst Transaction Length Equal To Rx FIFO Threshold . . . . .	838
20.3.14	Reset Signals . . . . .	839
20.4	UART Register Description . . . . .	841
20.4.1	Register Map Summary . . . . .	841
20.4.2	Receive Buffer Register. . . . .	845
20.4.3	Transmit Holding Register. . . . .	846
20.4.4	divisor latch low. . . . .	846
20.4.5	interrupt enable register . . . . .	847

20.4.6	divisor latch high . . . . .	848
20.4.7	interrupt identity register . . . . .	849
20.4.8	FIFO control register . . . . .	849
20.4.9	line control register . . . . .	850
20.4.10	modem control register . . . . .	852
20.4.11	line status register . . . . .	854
20.4.12	modem status register . . . . .	856
20.4.13	scratch register . . . . .	858
20.4.14	Low Power Divisor Latch Low . . . . .	858
20.4.15	Low Power Divisor Latch Low . . . . .	859
20.4.16	shadow receive buffer registers . . . . .	860
20.4.17	shadow transmit holding registers . . . . .	861
20.4.18	FIFO access register . . . . .	861
20.4.19	transmit FIFO read . . . . .	862
20.4.20	receiver FIFO write . . . . .	862
20.4.21	uart status register . . . . .	864
20.4.22	transmit FIFO level . . . . .	865
20.4.23	receive FIFO level . . . . .	865
20.4.24	software reset register . . . . .	865
20.4.25	shadow request to send . . . . .	866
20.4.26	shadow request to send . . . . .	867
20.4.27	shadow break control . . . . .	867
20.4.28	shadow FIFO enable . . . . .	868
20.4.29	shadow receiver trigger . . . . .	868
20.4.30	shadow transmitter trigger . . . . .	869
20.4.31	halt Tx . . . . .	869
20.4.32	dma software acknowledge . . . . .	870
20.4.33	Transceiver Control Register . . . . .	871
20.4.34	Driver Output Enable Register . . . . .	871

20.4.35	Receiver Output Enable Register . . . . .	872
20.4.36	Driver Output Enable Timing Register . . . . .	872
20.4.37	TurnAround Timing Register . . . . .	873
20.4.38	Divisor Latch Fraction Register . . . . .	873
20.4.39	Receive Address Register . . . . .	874
20.4.40	Transmit Address Register . . . . .	874
20.4.41	Line Extended Control Register . . . . .	875
20.4.42	Component Parameter Register . . . . .	876
20.4.43	UART Component Version . . . . .	877
20.4.44	Component Type Register . . . . .	877
<b>21</b>	<b>PWM/TIMER</b>	<b>878</b>
21.1	Overview . . . . .	878
21.2	Features . . . . .	881
21.3	PWM Operation . . . . .	882
21.3.1	Prescaler and Divider . . . . .	882
21.3.2	Basic Timer Operation . . . . .	882
21.3.3	Auto-reload and Double Buffering . . . . .	884
21.3.4	Timer Operation . . . . .	885
21.3.5	Initialize Timer (Setting Manual-Up Data and Inverter) . . . . .	886
21.3.6	PWM (Pulse Width Modulation) . . . . .	886
21.3.7	Output Level Control . . . . .	887
21.3.8	Dead Zone Generator . . . . .	888
21.4	I/O Description . . . . .	889
21.5	PWM Register Description . . . . .	890
21.5.1	Register Map Summary . . . . .	890
21.5.2	T0CFG0 . . . . .	891
21.5.3	T0CFG1 . . . . .	891
21.5.4	T0CON . . . . .	891

21.5.5	T0CMPB . . . . .	892
21.5.6	T0CNTO . . . . .	892
21.5.7	T0INT_CSTAT . . . . .	893
21.5.8	T0CFG0 . . . . .	893
21.5.9	T0CFG1 . . . . .	893
21.5.10	T0CON . . . . .	894
21.5.11	T0CMPB . . . . .	894
21.5.12	T0CNTO . . . . .	895
21.5.13	T0INT_CSTAT . . . . .	895
21.5.14	T0CFG0 . . . . .	895
21.5.15	T0CFG1 . . . . .	896
21.5.16	T0CON . . . . .	896
21.5.17	T0CMPB . . . . .	897
21.5.18	T0CNTO . . . . .	897
21.5.19	T0INT_CSTAT . . . . .	897
21.5.20	T0CFG0 . . . . .	898
21.5.21	T0CFG1 . . . . .	898
21.5.22	T0CON . . . . .	898
21.5.23	T0CMPB . . . . .	899
21.5.24	T0CNTO . . . . .	899
21.5.25	T0INT_CSTAT . . . . .	900
<b>22</b>	<b>WDT</b>	<b>901</b>
22.1	Overview . . . . .	901
22.2	Functional Description . . . . .	902
22.2.1	Watchdog Timer Operation . . . . .	902
22.2.2	WTDAT & WTCNT . . . . .	902
22.2.3	Special Function Register . . . . .	903
22.2.3.1	Memory map . . . . .	903

22.2.3.2	Watchdog timer control(WTCON) register . . . . .	903
22.2.3.3	Watchdog timer data(WTDAT) register . . . . .	903
22.2.3.4	Watchdog timer count(WTCNT) register . . . . .	903
22.2.3.5	Watchdog timer interrupt(WTCLRINT) register . . . . .	903
22.3	WDT Register Description . . . . .	904
22.3.1	Register Map Summary . . . . .	904
22.3.2	WTCON . . . . .	905
22.3.3	WTDAT . . . . .	905
22.3.4	WTCNT . . . . .	905
22.3.5	WTCLRINT . . . . .	906

# List of Tables

3.1 APB Peripheral Address Map . . . . .	16
5.1 PLL PHY Control Signals for Setting . . . . .	29
6.1 Clock Sources . . . . .	35
7.1 maximum packet size per endpoint . . . . .	56
10.1 Horizontal & Vertical Timing Symbols . . . . .	343
10.2 Register Setting for ITU-R BT.601 . . . . .	344
10.3 Embedded Sync Code . . . . .	346
10.4 Register Settings for ITU-R BT.656 . . . . .	347
10.5 VIP Data Order . . . . .	347
10.6 Recommend Setting for Video Input Port . . . . .	348
10.7 Registers for Scaling . . . . .	350
10.8 Interrupt registers . . . . .	350
11.1 Supported Video Standards . . . . .	367
14.1 Timing Charactoristics . . . . .	419
14.2 Timing Charactoristics Case1 . . . . .	421
14.3 Timing Charactoristics Case2 . . . . .	423
14.4 Pin Descriptions . . . . .	425
15.1 I2C/SMBus Definition of Bits in First Byte . . . . .	443
15.2 SMBus Bus Protocols Usage in DW_apb_i2c . . . . .	469
15.3 Derivation of SMBus ARP Command Through TxFIFO Commands . . . . .	473
15.4 Derivation of I2C Timing Parameters from *CNT Registers . . . . .	478
15.5 ic_clk in Relation to High and Low Counts When IC_CLK_FREQ_OPTIMIZATION = 0 . . . . .	482

15.6 ic_clk in Relation to High and Low Counts When IC_CLK_FREQ_OPTIMIZATION = 1 . . . . .	483
15.7 Minimum IC_CLK Frequency in Slave Mode When IC_CLK_FREQ_OPTIMIZATION=1 . . . . .	484
15.8 ic_clk in relation to High and Low Counts when IC_ULTRA_FAST_MODE=1 . . . . .	485
15.9 Maximum Values for IC_SDA_RX_HOLD . . . . .	489
15.10 Clearing and Setting of Interrupt Registers . . . . .	500
15.11 Top Level Parameters . . . . .	507
15.12 I2C Version 3.0 Features Parameters . . . . .	508
15.13 SMBus Features Parameters . . . . .	509
15.14 I2C Version 6.0 Features Parameters . . . . .	509
15.15 Interrupt Signals . . . . .	517
15.16 I2C Interface (Master/Slave) Signals . . . . .	520
15.17 APB Slave Interface Signals . . . . .	522
15.18 DMA Interface Signals . . . . .	524
15.19 SMBus Interface Signals . . . . .	525
15.20 I2C Debug Signals . . . . .	527
16.1 DES0 . . . . .	622
16.2 DES1 . . . . .	622
16.3 DES2 . . . . .	623
16.4 DES3 . . . . .	623
16.5 DES4 . . . . .	623
16.6 DES5 . . . . .	623
16.7 DES6 . . . . .	623
16.8 DES7 . . . . .	624
17.1 Key Features of the DMA Controllers . . . . .	654
17.2 Peripheral DMAC0 Request ID . . . . .	655
17.3 Peripheral DMAC1 Request ID . . . . .	655

17.4 Instruction Syntax Summary . . . . .	659
17.5 DMAMOV CCR Argument Description and the Default Values . . . . .	661
18.1 Pull-Up Resister Current . . . . .	702
19.1 ADDR_L Decode in Enhanced SPI Mode . . . . .	752
19.2 DMA Transmit Data Level (DMATDL) Decode Value . . . . .	759
19.3 DMA Receive Data Level (DMARDL) Decode Value . . . . .	759
20.1 Divisor Latch Fractional Values . . . . .	806
20.2 Narrow Pulse Exceptions . . . . .	810
20.3 Narrow Pulse Exceptions . . . . .	818
20.4 DW_apb_uart/DW_ahb_dmac Settings for Deadlock When Transaction Less Than Rx FIFO Threshold	837
21.1 Minimum and Maximum Resolution based on Prescaler and Clock Divider Values . . . . .	882
22.1 WDT Mamory map . . . . .	903

# 1 Product Overview

## 1.1 Introduction

DRONE\_SOC is a system-on-a-chip (SoC) based on the 64-bit RISC processor. Designed with the 55 nm low power process, features of DRONE\_SOC include:

- Dual-core RISC-V CPU
- Highest memory bandwidth
- HD(1280x720) 30 frame video encoding hardware
- High-speed interfaces such as eMMC4.5 and USB 2.0 Device

## 1.2 Features

- 55 nm Process Technology
- 529 ball PBGA Package, 0.8 mm Ball Pitch, 19X19 mm Body size
- RISC-V Dual-Core CPU @ 200MHz
- Supports various memory
- HD Multi Format Video Encoder
- Supports 1280x720 Scaler
- Supports ITUR.BT 656 Parallel Video Interface
- 32bit LPDDR1 200 MHz and Max Capacity 512MB
- Supports 2-ch SD/MMC, 4-ch UARTs, 4-ch USART, 2 32-ch DMAs, Interrupt Controller, RTC
- Supports 12-ch I2C, 3-ch SPI, 5-ch QSPI, 12-ch PWM, 8-ch Timer, 1 WDT and GPIOs
- Supports 4-ch 12-bit ADC
- Supports USB 2.0 Device Only
- Supports Security functions AES
- Supports two boot modes including SD(eMMC) and SPI

### 1.3 Block Diagram

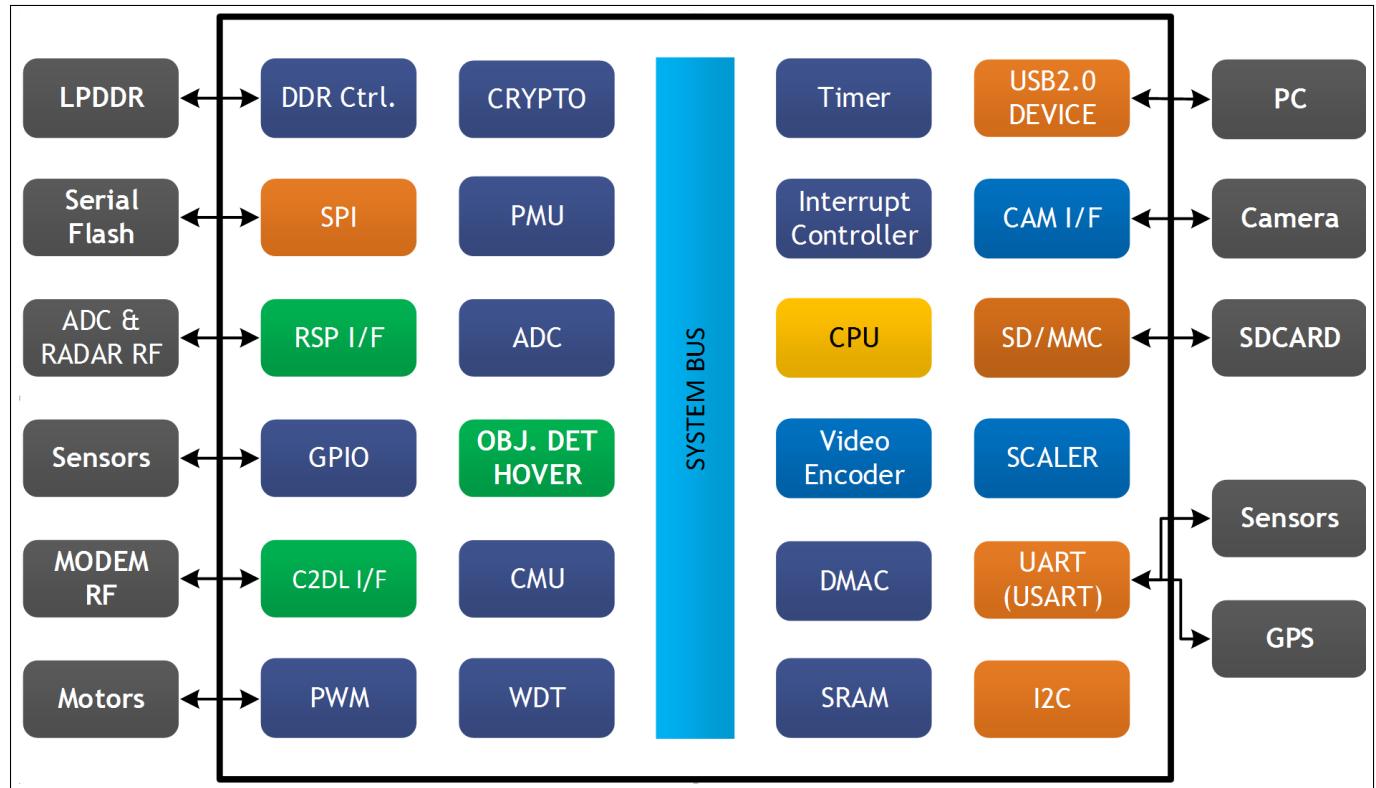


Figure 1.1: DRONE\_SOC Block Diagram

## 1.4 Brief Functional Specification

### 1.4.1 SYSREG

- There is one SYSREG IPs for DRONE\_SOC
  - SYSREG\_SYS

### 1.4.2 PLL

- Total 2 PLL for DRONE\_SOC
  - PLL\_0 (for SYSTEM)  
Frequency : from 24M to 800MHz
  - PLL\_1 (for RISC-V)  
Frequency : from 24M to 800MHz
- Frequency is changed by Programmable Divider(REFDIV, FBDIV, GV)
- FSM based MUX select change for stable system.
- Support output OSCCLK
- Support PLL power off (PLL RESET)

### 1.4.3 CMU

- Supports mode for reset release without sync-clock.
- Supports glitch-less mux for bus clock divider.

### 1.4.4 USB20OTG

- Complies with the On-The-Go Supplement to the USB 2.0 Specification (Revision 3.10a)
- Operates in High-Speed (480 Mbps), Full-Speed (12 Mbps) and Low-Speed (1.5 Mbps, Host only) modes
- Supports UTMI+ Level 3 interface (Revision 1.0)
- Supports Session Request Protocol (SRP) and Host Negotiation Protocol (HNP)
- 1 Control Endpoint 0 for control transfer
- 16 Device Mode Endpoints including Control Endpoint 0
- Programmable endpoint type: Bulk, Isochronous, or Interrupt
- Programmable IN/ OUT direction
- Supports 16 Host channels

#### 1.4.5 DDRC

- 512MB LPDDR1 at 200MHz support

#### 1.4.6 HOVER\_DMA

- Supports 60K x 60K pixels/frame
- Supports 60K byte stride
- Supports non-aligned base address
- Generating video signals for HOVER Format
- Input address/stride FIFO (4depth) If the input FIFO is empty, it will display using the last used address
- Output address FIFO (4depth) for software implementation convenience If an address is no longer used, the address is returned via this output FIFO
- Supports user programmable interrupt
- 32bit APB BUS for SFR
- 128 bit AXI4 BUS for 2D image reading 32bit Address

#### 1.4.7 VIP

- Supports ITU-BT601, ITU-BT656(YUV 422 & RAW).
- Clock, HSync, VSync, Field, 8bit data ports.
- Maximum 8192x8192 image Supports.
- Supports 3-Plane YUV 4:2:0, 4:2:2, 4:4:4, LinearYUYV, 8Bit RAW
- Clipping and Scale Down.
- Vertical and Operation Done Interrupt.
- 1 Inputs from External CIS.

#### 1.4.8 Multi-Format Video Codec

- Full HD multi-standard video IP for HDTVs, HD set-top boxes, and HD DVD players.
- Compressed video in a format of H.264 BP/MP/HP, VC-1 SP/MP/AP, MPEG-1/2, MPEG-4 SP/ASP, H.263P3, VP8,Theora, AVS, RV-8/9/10, and JPEG (max. 8192 x 8192).
- Performs H.264, MPEG-4, and H.263 encoding up to Full-HD 1920 x 1088 (max. 8192 x 8192 JPEG) resolution.
- The VPU can perform simultaneous multiple real time encoding, decoding, or both encoding and decoding of different format video streams at multiple resolutions.
- The VPU contains a 16-bit DSP called BIT processor.
- The BIT processor communicates with a host CPU through a host interface and controls the other sub-blocks of the VPU.

- The host CPU require slow resources under 1 MIPS, because all of the functions such as bit stream parsing, video hardware sub-blocks control and error resilience are implemented in the BIT processor. Moreover it is designed to optimally share most of the sub-blocks that are used in common for video processing, which contributes to the ultra low power and low gate count.
- It is connected with a host CPU system via 32-bit AMBA 3 APB bus for system control and 128-bit AMBA3 AXI for data.
- There are two 128-bit AXI buses: primary and secondary.
- The secondary bus can be connected to on-chip memories to achieve high performance.

#### 1.4.9 Scaler

- Source/Destination Image
- Format: Separated YUV Format (420, 422, 444), Interleaved UV
- Size: (8 to 4096) x (8 to 4096) (Width is set as a multiple of eight).
- Upscale Ratio:  $8 \times 8 \rightarrow 4096 \times 4096$
- Downscale Ratio:  $4096 \times 4096 \rightarrow 8 \times 8$
- Lowpass filter available after Upscale or before Downscale.
- Horizontal 5-Tab Filter: Coefficients 64 Sets.
- Vertical 3-Tab Filter: Coefficients 32 Sets (For Frequency Response, refer to Operation Item).

#### 1.4.10 Crypto Engine

- Big-endian Encryption & Decryption
- Supports DMA Interface
- Supports AES ECB, CBC, CTR -128,192, 256 Mode
- Supports DES ECB, CBC -64 Mode
- Supports 3DES -64 Mode
- Supports HASH Mode (SHA1, MD5)
- Supports input share Mode (AES & HASH)
- Supports AES and HASH working at the same time (refer in the following figure)

### 1.4.11 ADC

- 28nm Low Power CMOS Process
- Resolution: 12-bit
- Maximum Conversion rate (Fs)
  - 1MSPS (main clock: 6MHz / sampling clock: 1MHz)
- Power consumption:
  - 1.0 mW ( Fs = 1MSPS ) @ Normal operation mode Typ
  - 0.005 mW @ Power down mode Typ
- Input range: AGND ~ VREF (0 ~ AVDD18A1 at Typical Condition)
- Input frequency : DC ~ 100 kHz ( @ Fs = 1MSPS )
- Digital output: CMOS Level (0 ~ AVDD10)
- Operation temperature range (ambient) : -25°C ~ 85°C

### 1.4.12 I2C

- 12-ch I2C bus controller
- Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; master can operate as master-transmitter or as master-receiver
- Serial, 8-bit oriented, bi-directional data transfers can be made at up to 100kbit/s in the Standard-mode, up to 400kbit/s in the Fast-mode and up to 1000kbit/s in the Fast-mode-plus
- The number of ICs that can be connected to the same bus is limited only by a maximum bus capacitance of 400pF
- High speed mode are not supported

### 1.4.13 SD/MMC Controller

- 3 ports SD/HD-MMC interfaces
- 256-word FIFO for Tx/Rx
- Secure Digital Memory (SD mem- version 3.0)
- Secure Digital I/O (SDIO - version 3.0)
- Consumer Electronics Advanced Transport Architecture (CE-ATA - version 1.1)
- Multimedia Cards (MMC - version 4.41, **eMMC 5.0**)
- Supports 8bit DDR mode up to **200MHz**
- Supports PIO and DMA mode data transfer

#### 1.4.14 DMA

- Two 8-channel non-secure PDMA

#### 1.4.15 GPIO

- Programmable Pull-Up Control
- Edge/Level Detect
- Supports programmable Pull-Up resistance.
- Supports four event detection modes
  - Rising Edge Detection
  - Falling Edge Detection
  - Low Level Detection
  - High Level Detection
- The number of GPIOs : 144
- Supports secure / non-secure mode.
- 3-ch SPI, 5-ch QSPI controller
- Supports the Motorola SPI protocol, National Semiconductor Microwire and Texas Instruments SSP
- Supports 8-bit/16-bit/32-bit bus interface
- Master & Slave mode
  - SPI : Master & Slave mode(SPI1/SPI2)
  - QSPI : Master Only
- Supports Dual Data Rate (DDR) - when Dual/Quad mode of operation
- Supports eXecute-In-Place(XIP) - when Dual/Quad mode of operation
- Supports DMA
- FIFO depth - the master has 32 transmit / receive FIFOs, and the slave has 8 transmit / receive FIFOs.
- Programmable delay on the sample time of the received serial data bit (rxd), when configured in Master Mode; enables programmable control of routing delays resulting in higher serial data-bit rates.
- Inform the system that a receive FIFO (over-run/under-run) has occurred
- Inform the system that a multi-master contention has occurred
- Maximum SPI CORE frequency is 200 MHz
  - Master Mode : Max 100MHz
  - Slave Mode : 5MHz (8MHz using DMA)

#### 1.4.16 UART & ISO7816 Sim Card Interface

- 9-bit serial data support
- False start bit detection
- Programmable fractional baud rate support
- Multi-drop RS485 interface support
- Configurable parameters for the following:
  - APB data bus widths of 8, 16 and 32
  - Additional DMA interface signals for compatibility with DesignWare DMA interface
  - DMA interface signal polarity
  - Transmit and receive FIFO depths of 0, 16, 32, 64, 128, 256, 512, 1024, 2048
  - Internal or external FIFO (RAM) selection
  - Use of two clocks—pclk and sclk—instead of just pclk
  - IrDA 1.0 SIR mode support with up to 115.2 Kbaud data rate and a pulse duration (width) as specified in the IrDA physical layer specification: width = 3/16  $\mu$ bit period
  - IrDA 1.0 SIR low-power reception capabilities
  - Baud clock reference output signal
  - Clock gate enable output(s) used to indicate that the TX and RX pipeline is clear (no data) and no activity has occurred for more than one character time, so that clocks can be gated
  - FIFO access mode—for FIFO testing—enabling the master to write to the receive FIFO and read from the transmit FIFO
  - Additional FIFO status registers
  - Shadow registers to reduce software overhead and also include a software programmable reset
  - Auto Flow Control mode, as specified in the 16750 standard
  - Loopback mode that enables greater testing of Modem Control and Auto Flow Control features (Loopback support in IrDA SIR mode is available)
  - Transmitter Holding Register Empty (THRE) interrupt mode
  - Busy functionality
- Ability to set some configuration parameters during instantiation
- Configuration identification registers present
- Functionality based on the 16550 industry standard
  - Programmable character properties, such as:
    - \* Number of data bits per character (5-8)
    - \* Optional parity bit (with odd, even select or Stick Parity)
    - \* Number of stop bits (1, 1.5 or 2)
  - Line break generation and detection
  - DMA signaling with two programmable modes
  - Prioritized interrupt identification
- Programmable FIFO enable/disable
- Programmable serial data baud rate as calculated by the following: baud rate = (serial clock frequency)/(16 $\mu$ divisor)
- External read enable signal for RAM wake-up when using external RAMs
- Modem and status lines are independently controlled
- Separate system resets for each clock domain to prevent metastability

### 1.4.17 PWM

- Four 32-bit Timers
- Four 8-bit Clock Prescalers that provide first level of division for the TCLK or the Oscillator Clock, and four Clock Dividers and Multiplexers that provide second level of division for the Prescaler clock
- To use Oscillator Clock as source clock, User must receive USE\_OSC signal from SYSREG outside PWM IP
- Programmable Clock Select Logic for individual PWM Channels
- Four Independent PWM Channels with Programmable Duty Control and Polarity
- Static Configuration: PWM is stopped
- Dynamic Configuration: PWM is running
- Auto-Reload and One-Shot Pulse Mode
- Dead Zone Generator on all PWM Outputs
- Level Interrupt Generation

### 1.4.18 WDT

- Non-secure WDT : Normal interval timer mode with interrupt request
- Secure WDT : Internal reset signal is activated when the timer count value reaches 0 (time-out).
- Level-triggered interrupt mechanism

# 2 Mechanical Dimension

## 2.1 Mechanical Dimension

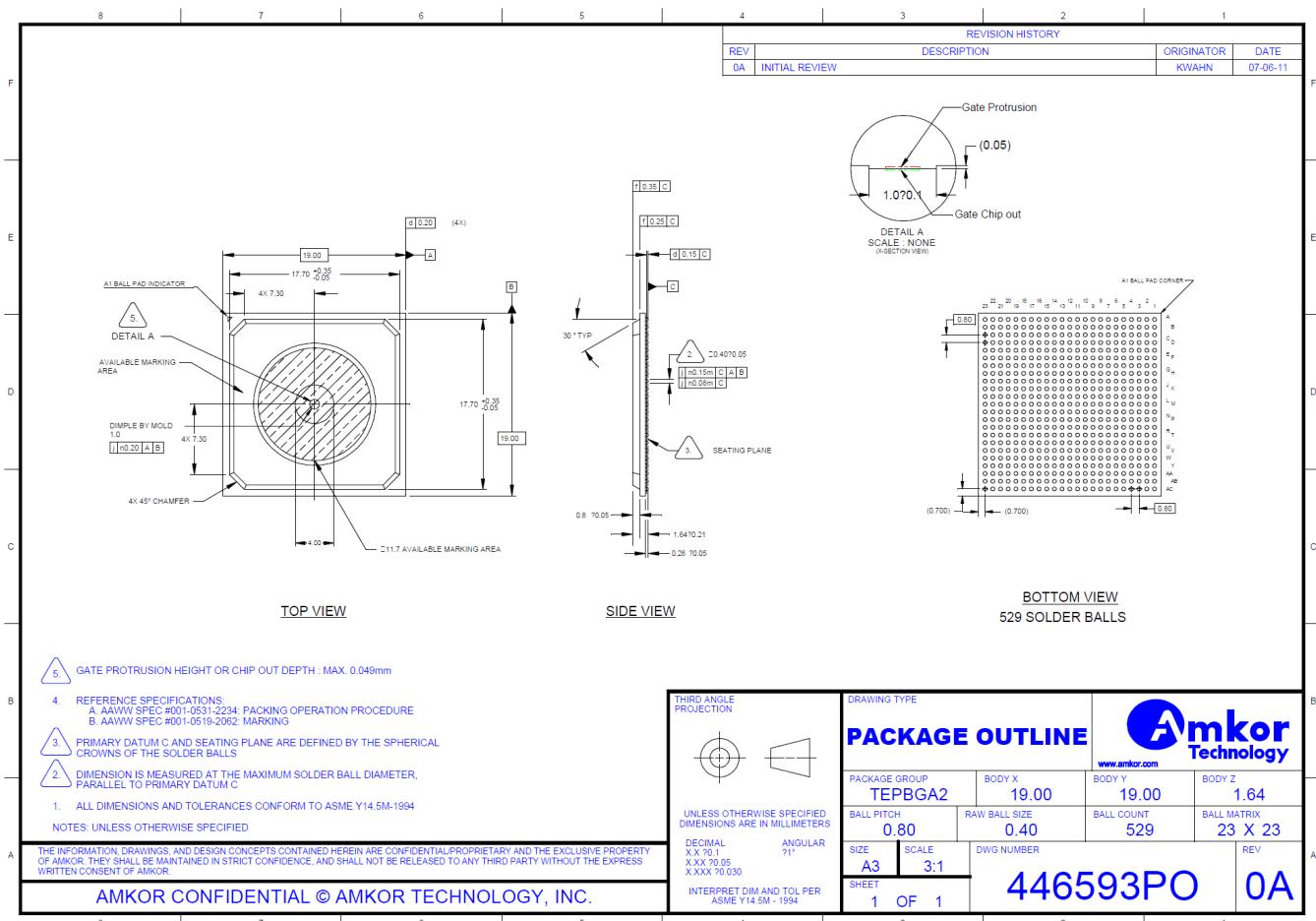


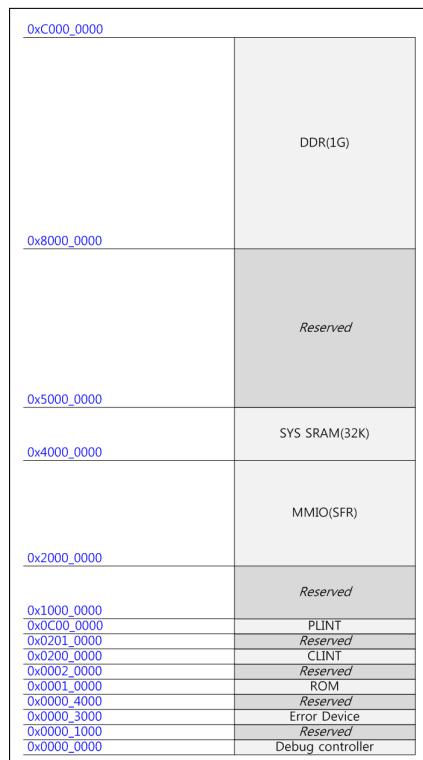
Figure 2.1: Mechanical dimension

# 3 Address Map

## 3.1 Overview

The DRONE memory map includes these regions:

- DRAM (Dynamic Random Access Memory): Represents the external RAM.
- SFRs (Special Function Registers): Represents the SFRs
- ROM/RAM/MCUSTATIC : Represents the internal memories



**Figure 3.1:** DRONE\_SOC Memory Map Block Diagram

## 3.2 SFR Address Map

Instance	Base Address
SYSCON[0]	0x2000_0000
PLL[0]	0x2001_0000
PLL[1]	0x2002_0000
CMU_SYS[0]	0x2003_0000
SYSREG_SYS[0]	0x2004_0000
DDRC[0]	0x2005_0000
AXIM_DMA[0]	0x2020_0000
AXIM_DMA[1]	0x2021_0000
AXIM_CPUSFR[0]	0x2022_0000
AXIM_CPUTMEM[0]	0x2023_0000
AXIM_VIP[0]	0x2024_0000
AXIM_USB[0]	0x2025_0000
AXIM_HOVER[0]	0x2026_0000
AXIM_SCALER[0]	0x2027_0000
AXIM_CODA[0]	0x2028_0000
AXIM_OSDMMC[0]	0x2029_0000
AXIM_OSDMMC[1]	0x202A_0000
DMA_DATA_BUS_CFG[0]	0x2030_0000
VIP_DATA_BUS_CFG[0]	0x2031_0000
CODA_DATA_BUS_CFG[0]	0x2032_0000
MAIN_BUS_CFG[0]	0x2033_0000
CFG_BUS_CFG[0]	0x2034_0000
VIP[0]	0x2040_0000
SCALER[0]	0x2041_0000
CODA[0]	0x2042_0000
RSP[0]	0x2043_0000
CRYPTO[0]	0x2044_0000
DMA[0]	0x2045_0000
DMA[1]	0x2046_0000
HOVER[0]_HOVER	0x2047_0000
HOVER[0]_DMA	0x2048_0000
I2C[0]	0x2060_0000
I2C[1]	0x2061_0000
I2C[2]	0x2062_0000
I2C[3]	0x2063_0000
I2C[4]	0x2064_0000
I2C[5]	0x2065_0000
I2C[6]	0x2066_0000
I2C[7]	0x2067_0000
I2C[8]	0x2068_0000
I2C[9]	0x2069_0000
I2C[10]	0x206A_0000
I2C[11]	0x206B_0000
ADC[0]	0x206C_0000
WDT[0]	0x206D_0000
GPIO[0]	0x2070_0000
GPIO[1]	0x2071_0000
GPIO[2]	0x2072_0000
GPIO[3]	0x2073_0000
GPIO[4]	0x2074_0000
GPIO[5]	0x2075_0000
GPIO[6]	0x2076_0000

Instance	Base Address
GPIO[7]	0x2077_0000
SPI[0]	0x2080_0000
SPI[1]	0x2081_0000
SPI[2]	0x2082_0000
QSPI[0]	0x2083_0000
QSPI[1]	0x2084_0000
QSPI[2]	0x2085_0000
QSPI[3]	0x2086_0000
QSPI[4]	0x2087_0000
UART[0]	0x2088_0000
UART[1]	0x2089_0000
USART[0]	0x208A_0000
USART[1]	0x208B_0000
USART[2]	0x208C_0000
USART[3]	0x208D_0000
PWM[0]	0x208E_0000
PWM[1]	0x208F_0000
PWM[2]	0x2090_0000
TIMER[0]	0x2091_0000
TIMER[1]	0x2092_0000
OSDMMC[0]	0x2093_0000
OSDMMC[1]	0x2094_0000
C2DL[0]	0x20C0_0000
USB[0]	0x20D0_0000

**Table 3.1:** APB Peripheral Address Map

# 4 SYSREG

## 4.1 Overview

SYSREG generates control signals for IPs. Such as HOVER, RSP, DDRC, SPI, USART, PWM, TIMER and USB.

## 4.2 Features

- There is one SYSREG IPs for DRONE\_SOC
  - SYSREG\_SYS

## 4.3 SYSREG\_SYS Register Description

### 4.3.1 Register Map Summary

- Base Address: 20040000(SYSREG\_SYS)

Register	Offset	Description	Reset Value
SPARE0	0x00	Spare Control Register 0	0x0000_0000
SPARE1	0x04	Spare Control Register 1	0x0000_0000
SPARE2	0x08	Spare Control Register 2	0x0000_0000
SPARE3	0x0C	Spare Control Register 3	0x0000_0000
SPARE4	0x10	Spare Control Input Register 0	0x0000_0000
HOVER_0_CTRL0	0x30	HOVER_0 Control Register	0x0000_0000
RSP_0_CTRL0	0x34	RSP_0 Control Register	0x0000_0000
DDRC_0_CTRL0	0x38	DDRC_0 Control Register	0x0000_0000
SPI_0_CTRL0	0x40	SPI_0 Control Register	0x0000_0000
SPI_1_CTRL0	0x44	SPI_1 Control Register	0x0000_0000
SPI_2_CTRL0	0x48	SPI_2 Control Register	0x0000_0000
USART_0_CTRL0	0x60	USART_0 Control Register	0x0000_0000
USART_1_CTRL0	0x64	USART_1 Control Register	0x0000_0000
USART_2_CTRL0	0x68	USART_2 Control Register	0x0000_0000
USART_3_CTRL0	0x6C	USART_3 Control Register	0x0000_0000
PWM_0_CTRL0	0x70	PWM_0 Control Register	0x0000_0000
PWM_1_CTRL0	0x74	PWM_1 Control Register	0x0000_0000
PWM_2_CTRL0	0x78	PWM_2 Control Register	0x0000_0000
TIMER_0_CTRL0	0x80	TIMER_0 Control Register	0x0000_0000
TIMER_1_CTRL0	0x84	TIMER_1 Control Register	0x0000_0000
USB_PHY_CFG_0	0x90	USB_0 PHY Config0	0x0000_1A41
USB_PHY_CFG_1	0x94	USB_0 PHY Config1	0x0020_DB91
USB_PHY_CFG_2	0x98	USB_0 PHY Config2	0x0000_0004
DMA_BOOT_CFG	0x100	DMA boot manager ns	0x0000_0003
DMA_0_CTRL0	0x110	DMA_0 Control Register0	0xFFFF_FFFF
DMA_1_CTRL0	0x114	DMA_0 Control Register1	0xFFFF_FFFF
DMA_0_CTRL0	0x120	DMA_1 Control Register0	0xFFFF_FFFF
DMA_0_CTRL0	0x124	DMA_1 Control Register1	0xFFFF_FFFF
USB_PHY_CFG_3	0x128	USB_0 PHY Config3	0x0000_0000

#### 4.3.2 SPARE0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SPARE_OUT_0	[31:0]	R/W	Spare Control Register 0	0

#### 4.3.3 SPARE1

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SPARE_OUT_1	[31:0]	R/W	Spare Control Register 1	0

#### 4.3.4 SPARE2

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SPARE_OUT_2	[31:0]	R/W	Spare Control Register 2	0

#### 4.3.5 SPARE3

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SPARE_OUT_3	[31:0]	R/W	Spare Control Register 3	0

#### 4.3.6 SPARE4

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SPARE_IN_0	[31:0]	R	Spare Control Input Register 0	0

#### 4.3.7 HOVER\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
HOVER_0__SYSREG__ _HOVER_CLR	[0]	R/W	Parameter Setting Clear	0

#### 4.3.8 RSP\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
RSP_0__SYSREG__ CLR	[0]	R/W	Parameter Setting Clear(SW Reset)	0

#### 4.3.9 DDRC\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
DDRC_0__SYSREG__ INITDONE	[1]	R	DDRC Interrupt Done	0
DDRC_0__SYSREG__ INITOK	[0]	R	DDRC Interrupt OK	0

#### 4.3.10 SPI\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SPI_0__SYSREG__MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.11 SPI\_1\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SPI_1__SYSREG__MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.12 SPI\_2\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x48 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SPI_2__SYSREG__MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.13 USART\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x60 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
USART_0__SYSREG__MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.14 USART\_1\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x64 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
USART_1__SYSREG_MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.15 USART\_2\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x68 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
USART_2__SYSREG_MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.16 USART\_3\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x6C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
USART_3__SYSREG_MS_SEL	[0]	R/W	Master/Slave Select Register 0: Master, 1: Slave	0

#### 4.3.17 PWM\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x70 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
PWM_0__SYSREG_USE_OSCCLK0	[3]	R/W	USE OSCCLK for PWM0_0	0
PWM_0__SYSREG_USE_OSCCLK1	[2]	R/W	USE OSCCLK for PWM0_1	0
PWM_0__SYSREG_USE_OSCCLK2	[1]	R/W	USE OSCCLK for PWM0_2	0
PWM_0__SYSREG_USE_OSCCLK3	[0]	R/W	USE OSCCLK for PWM0_3	0

#### 4.3.18 PWM\_1\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x74 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
PWM_1_SYSREG_USE_OSCCLK0	[3]	R/W	USE OSCCLK for PWM1_0	0
PWM_1_SYSREG_USE_OSCCLK1	[2]	R/W	USE OSCCLK for PWM1_1	0
PWM_1_SYSREG_USE_OSCCLK2	[1]	R/W	USE OSCCLK for PWM1_2	0
PWM_1_SYSREG_USE_OSCCLK3	[0]	R/W	USE OSCCLK for PWM1_3	0

#### 4.3.19 PWM\_2\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x78 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
PWM_2_SYSREG_USE_OSCCLK0	[3]	R/W	USE OSCCLK for PWM2_0	0
PWM_2_SYSREG_USE_OSCCLK1	[2]	R/W	USE OSCCLK for PWM2_1	0
PWM_2_SYSREG_USE_OSCCLK2	[1]	R/W	USE OSCCLK for PWM2_2	0
PWM_2_SYSREG_USE_OSCCLK3	[0]	R/W	USE OSCCLK for PWM2_3	0

#### 4.3.20 TIMER\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x80 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
TIMER_0_SYSREG_USE_OSCCLK0	[3]	R/W	USE OSCCLK for TIMERO_0	0
TIMER_0_SYSREG_USE_OSCCLK1	[2]	R/W	USE OSCCLK for TIMERO_1	0
TIMER_0_SYSREG_USE_OSCCLK2	[1]	R/W	USE OSCCLK for TIMERO_2	0
TIMER_0_SYSREG_USE_OSCCLK3	[0]	R/W	USE OSCCLK for TIMERO_3	0

#### 4.3.21 TIMER\_1\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x84 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
TIMER_1__SYSREG__USE_OSCCLK0	[3]	R/W	USE OSCCLK for TIMER1_0	0
TIMER_1__SYSREG__USE_OSCCLK1	[2]	R/W	USE OSCCLK for TIMER1_1	0
TIMER_1__SYSREG__USE_OSCCLK2	[1]	R/W	USE OSCCLK for TIMER1_2	0
TIMER_1__SYSREG__USE_OSCCLK3	[0]	R/W	USE OSCCLK for TIMER1_3	0

#### 4.3.22 USB\_PHY\_CFG\_0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x90 Reset Value = 0x0000\_1A41

Name	Bit	Type	Description	Reset Value
RSVD	[31:14]	-	Reserved	-
USB_0__SYSREG__usb_phy_set0	[13:9]	R/W	USB PHY configuratoin 0 [13] Commonn : Reserved [12:11] refclksel : refernece clock select [10:9] refclkdiv : reference clock divider value	0xD
USB_0__SYSREG__usb_phy_set1	[8:0]	R/W	USB PHY configuratoin 1 : Reserved	0x41

#### 4.3.23 USB\_PHY\_CFG\_1

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x94 Reset Value = 0x0020\_DB91

Name	Bit	Type	Description	Reset Value
RSVD	[31:28]	-	Reserved	-
USB_0__SYSREG__usb_phy_set2	[27:24]	R/W	USB PHY configuratoin 2 : Reserved	0x0
USB_0__SYSREG__usb_phy_set3	[23:0]	R/W	USB PHY configuratoin 3 : Reserved	0x20DB91

#### 4.3.24 USB\_PHY\_CFG\_2

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x98 Reset Value = 0x0000\_0004

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
USB_0__SYSREG_ss_scaledown_mode	[7:6]	R/W	Reserved	0
USB_0__SYSREG_VBUSVLDEXT	[5]	R/W	Reserved	0
USB_0__SYSREG_VBUSVLDEXTSEL	[4]	R/W	Reserved	0
USB_0__SYSREG_POR_ENB	[3]	R/W	USB PHY POR enable	0
USB_0__SYSREG_POR	[2]	R/W	USB PHY POR	1
USB_0__SYSREG_PORTRESET	[1]	R/W	Reserved	0
USB_0__SYSREG_prst_n	[0]	R/W	PHY reset of USB LINK	0

#### 4.3.25 DMA\_BOOT\_CFG

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x100 Reset Value = 0x0000\_0003

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
DMA_1__SYSREG_BOOT_MANAGER_NS	[1]	R/W	Reserved (Tie off signal )	1
DMA_0__SYSREG_BOOT_MANAGER_NS	[0]	R/W	Reserved (Tie off signal )	1

#### 4.3.26 DMA\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x110 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
DMA_0__SYSREG_BOOT_IRQ_NS	[31:0]	R/W	Reserved (Tie off signal )	0xFFFFFFFF

#### 4.3.27 DMA\_1\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x114 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
DMA_0__SYSREG__ BOOT_PERIPH_NS	[31:0]	R/W	Reserved (Tie off signal )	0xFFFFFFFF

#### 4.3.28 DMA\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x120 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
DMA_1__SYSREG__ BOOT_IRQ_NS	[31:0]	R/W	Reserved (Tie off signal )	0xFFFFFFFF

#### 4.3.29 DMA\_0\_CTRL0

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x124 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
DMA_1__SYSREG__ BOOT_PERIPH_NS	[31:0]	R/W	Reserved (Tie off signal )	0xFFFFFFFF

#### 4.3.30 USB\_PHY\_CFG\_3

- Base Address: 20040000(SYSREG\_SYS)
- Address = Base Address + 0x128 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
USB_0__SYSREG__ usb_phy_set4	[31:0]	R/W	[31:3] Reserved [2] bus reset of USB LINK [1:0] Reserved	0x0

# 5 PLL

## 5.1 Overview

Phase Locked Loop (hereinafter, PLL) generated higher frequency with input reference clock. PLL received a FIN input and generates a FOUT output clock from 25MHz to 800MHz. CMU received PLL's FOUT for generating each IP clocks.

In DRONE\_SOC, PLL PHY is wrapped by PLL wrapper. PLL wrapper consists of PLL PHY, CPUIF, MUX and FSM logics. MUX has 2 inputs, OSCCLK (crystal or oscillator clock) and PLL PHY's output. Reset value of MUX select is OSCCLK.

When user changes PLL FOUT Frequency, FSM logics control the MUX's select signal. So when PLL is in reset state for using new configuration, PLL wrapper outputs OSCCLK. When PLL FOUT is stabilized, PLL wrapper changes the MUX's select to PLL FOUT. The time that PLL wrapper output OSCCLK, is configurable with CPUIF.

## 5.2 Features

- Total 2 PLL for DRONE\_SOC
  - PLL\_0 (for SYSTEM)  
Frequency : from 24M to 800MHz
  - PLL\_1 (for RISC-V)  
Frequency : from 24M to 800MHz
- Frequency is changed by Programmable Divider(REFDIV, FBDIV, GV)
- FSM based MUX select change for stable system.
- Support output OSCCLK
- Support PLL power off (PLL RESET)

### 5.3 Block Diagram

PLL wrapper consists of PLL PHY, CPUIF, MUX and FSM logics. Following figure shows PLL wrapper block diagram.

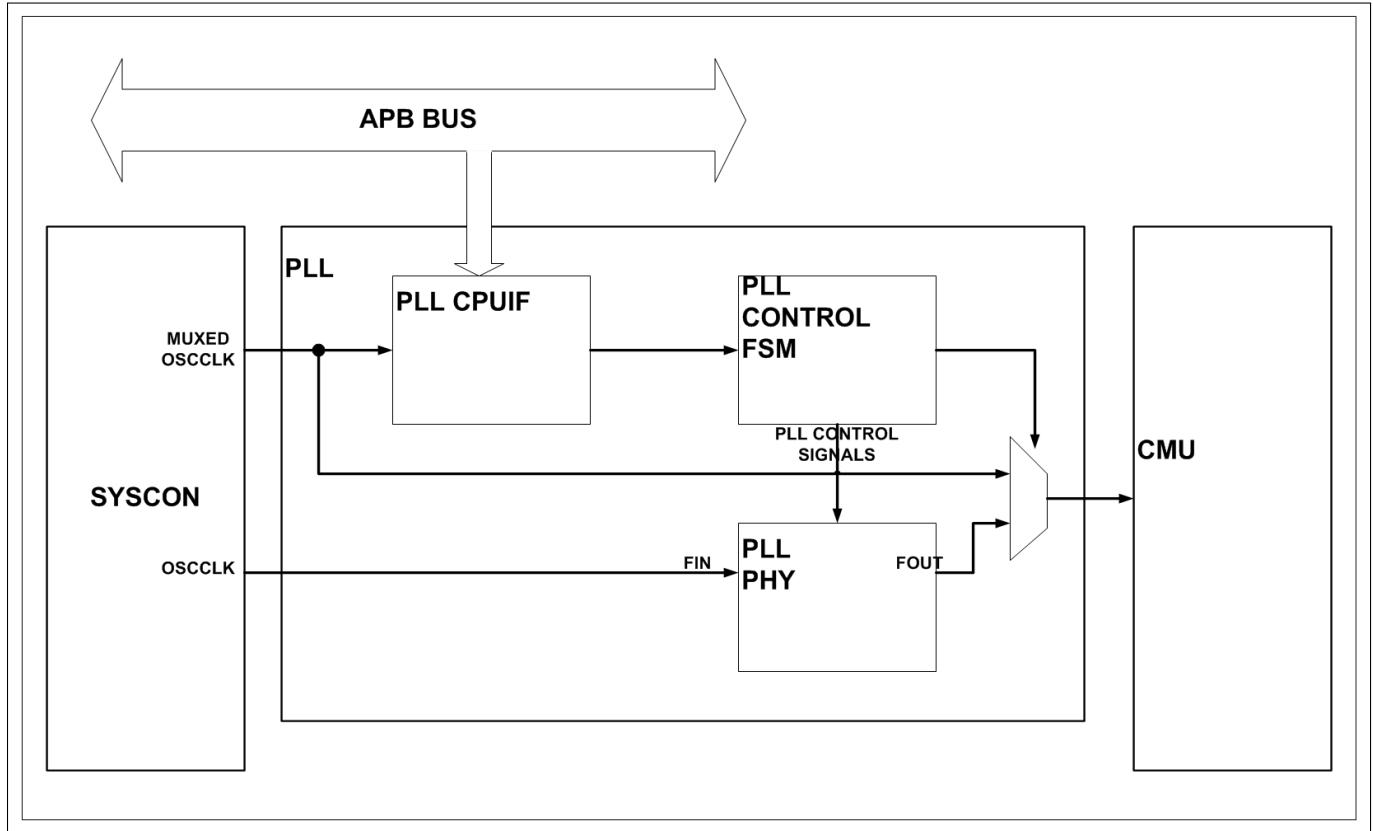
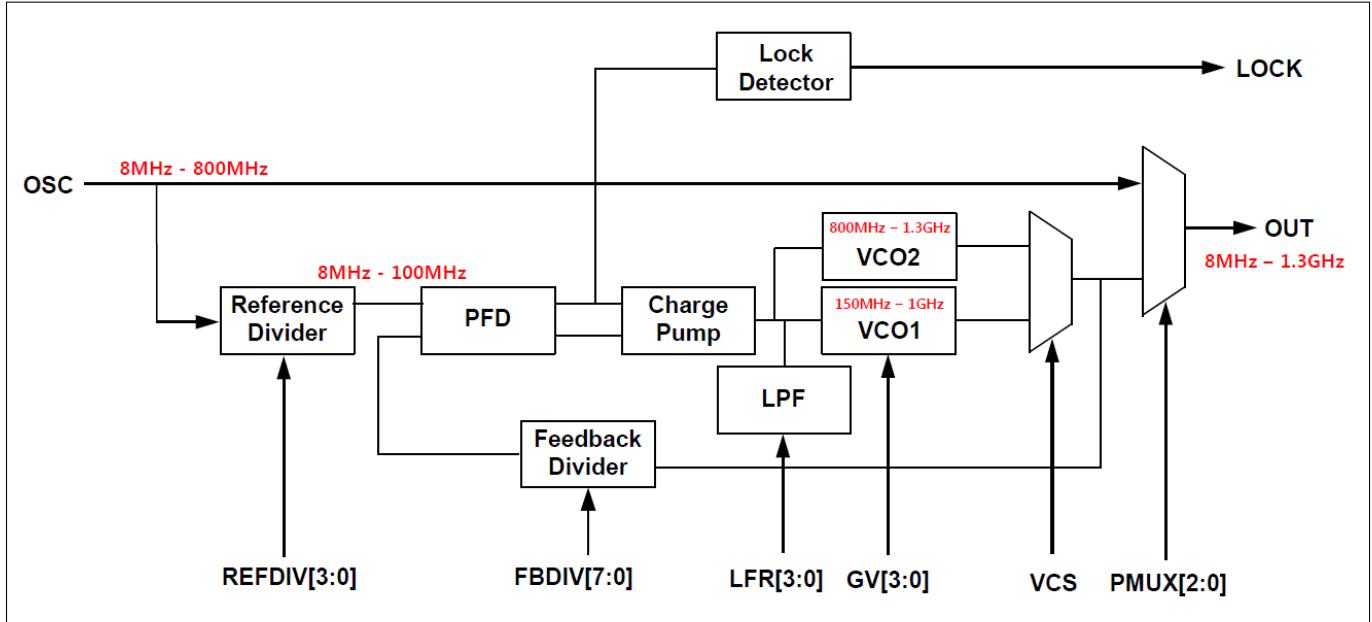


Figure 5.1: PLL Block Diagram

## 5.4 PLL Functional Description

Following figure shows PLL PHY block diagram.



**Figure 5.2:** PLL PHY Block Diagram

Following tables shows Setting value for Control PLL PHY.

I/O	Pin List	200MHz/1	300MHz/1	400MHz/1	300MHz/2	400MHz/2
IN	OSC	25MHz	25MHz	25MHz	25MHz	25MHz
IN	PD[0]	0	0	0	0	0
IN	VCS[0]	0	0	0	0	0
IN	LFR[3:0]	4	4	4	4	4
IN	FMUX[0]	0	0	0	0	0
IN	REFDIV[3:0]	0	0	0	0	0
IN	FBDIV[7:0]	8	12	16	12	16
IN	PMUX[2:0]	0	0	0	1	1
IN	GV[3:0]	2	2	4	2	4
OUT	LOCK					
OUT	OUT	200 MHz	300 MHz	400 MHz	150 MHz	200 MHz

**Table 5.1:** PLL PHY Control Signals for Setting

## 5.5 PLL Register Description

### 5.5.1 Register Map Summary

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)

Register	Offset	Description	Reset Value
PLLCTRL	0x00	PLL Control Register	0x0001_2110
PLLDBG0	0x04	PLL Debug Register 0	Undefined
PLLCNT0	0x10	PLL ST_RUN0 MaxCount Register	0x0000_0005
PLLCNT1	0x14	PLL ST_RUN1 MaxCount Register	0x0000_0005
PLLCNT2	0x18	PLL ST_RUN2 MaxCount Register	0x0000_0005
PLLCNT3	0x1C	PLL ST_RUN3 MaxCount Register	0x0000_0064
PLLCFG0	0x20	PLL ST_RUN0 Config Register	0x0000_0000
PLLCFG1	0x30	PLL ST_RUN1 Config Register	0x0080_1004
PLLLOCKINT0	0x40	PLL LOCK Interrupt Control Register	0x0000_0000

### 5.5.2 PLLCTRL

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x00 Reset Value = 0x0001\_2110

Name	Bit	Type	Description	Reset Value
RSVD	[31:17]	-	Reserved	-
CLKSEL_DONE	[16]	R	When OSCCLK_MUXSEL is configed and mux selection is done, this register be HIGH.	1
UPDATECONFIG_DIRECTLY	[15]	W	DIRECTLY UPDATECONFIG	0
CLKEN_DEBUG	[14:13]	R	For debug only.	1
CurSt	[12:8]	R	Current State Register, for debug.	1
CLKOFF	[7]	R/W	0 : CLK ON , 1 : CLK OFF - User must not set HIGH when this PLL is used for MAIN/CPU/DDR clock .	0
LOCK	[6]	R	LOCK Signal	0
RUNDONE	[5]	R	CurSt != ST_IDLE && NextSt == ST_IDLE	0
IDLE	[4]	R	CurSt == ST_IDLE	1
OSCCLK_MUXSEL	[3]	R/W	0 : Output OSCCLK , 1 : PLL FOUT	0
DIRTYFLAG_CLR	[2]	W	Clear DIRTYFLAG for Default Config	0
DIRTYFLAG	[1]	R/W	Set DIRTYFLAG for Default Config	0
RUN_CHANGE	[0]	W	PLL Run Change Start. Write Only	0

### 5.5.3 PLLDBG0

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x04 Reset Value = Undefined

Name	Bit	Type	Description	Reset Value
StCount	[31:0]	R	For debug only.	:no reset

### 5.5.4 PLLCNT0

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x10 Reset Value = 0x0000\_0005

Name	Bit	Type	Description	Reset Value
RUN0_MAXCOUNT	[31:0]	R/W	PD=LINK_PD(0)	5

### 5.5.5 PLLCNT1

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x14 Reset Value = 0x0000\_0005

Name	Bit	Type	Description	Reset Value
RUN1_MAXCOUNT	[31:0]	R/W	PD=1	5

### 5.5.6 PLLCNT2

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x18 Reset Value = 0x0000\_0005

Name	Bit	Type	Description	Reset Value
RUN2_MAXCOUNT	[31:0]	R/W	PD=1	5

### 5.5.7 PLLCNT3

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x1C Reset Value = 0x0000\_0064

Name	Bit	Type	Description	Reset Value
RUN3_MAXCOUNT	[31:0]	R/W	PD=LINK_PD(0)	100

### 5.5.8 PLLCFG0

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
CPUIF_PD	[0]	R/W	Default Configuration. ST_RUN0 Configuration	0

### 5.5.9 PLLCFG1

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x30 Reset Value = 0x0080\_1004

Name	Bit	Type	Description	Reset Value
RSVD	[31:25]	-	Reserved	-
LFR	[24:21]	R/W	Reference Clock Range -> Please refer to the specification or contact us to determine if the required setting is feasible	4
FMUX	[20]	R/W	Reference Divider Enable 0: off, 1: on	0
REFDIV	[19:16]	R/W	Reference Divider 2 15 2: /2, 3: /3, 4: /4, ...., 14: /14, 15: /15	0
FBDIV	[15:8]	R/W	Feed Back Divider 2 255 2: /2, 3: /3, 4: /4, ...., 254: /254, 255: /255	16
VCS	[7]	R/W	VCO Selection 0: VCO1, 1: VCO2	0
PMUX	[6:4]	R/W	Post Divier 0 7 0 : Direct Clock, 1: /2, 2: /4, 3: /8, 4: /16, 5: /32, 6: /64, 7: Reference Clock(Bypass)	0
GV	[3:0]	R/W	PLL Frequency Range , Please refer to the specification or contact us to determine if the required setting is feasible	4

### 5.5.10 PLLLOCKINT0

- Base Address: 20010000(PLL\_0)
- Base Address: 20020000(PLL\_1)
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
RSVD	[7]	-	Reserved	-
RUNDONE_INTDISABLE	[6]	W	Write 1 : Disable the RUNDONE Interrupt.	0
RUNDONE_INTENB	[5]	R/W	Write 1 : Enable the RUNDONE Interrupt. Read : Enable Status of RUNDONE Interrupt.	0
RUNDONE_INTPEND_CLR	[4]	R/W	Write 1 : Clear the RUNDONE Interrupt pending. Read : Status of RUNDONE Interrupt pending.	0
RSVD	[3]	-	Reserved	-
LOCK_INTDISABLE	[2]	W	Write 1 : Disable the LOCK Interrupt.	0
LOCK_INTENB	[1]	R/W	Write 1 : Enable the LOCK Interrupt. Read : Enable Status of LOCK Interrupt.	0
LOCK_INTPEND_CLR	[0]	R/W	Write 1 : Clear the LOCK Interrupt pending. Read : Status of LOCK Interrupt pending.	0

# 6 CMU

## 6.1 Overview

Clock Management Unit(hereinafter, CMU) can generate divided clock for each IP and BUS. And CMU modules has Reset Controller(hereinafter, RSTCON) for generate reset for each IP and BUS. Each IP have clocking scheme which requires several different division ratio simultaneously. Therefore, each of IP Clock Generator supplies required clock to each IP. These IP Clock Generators uses the PLL clock or External Clock from PAD.

CMU modules also supports reset control functions. CMU modules has two type of reset.

First one is STOP\_AND\_GO reset. This reset controlled by CMU, not RSTCON. When user releases this reset, CMU automatically clock disable, and then reset release, and finally clock enable. So, resets of this type used for BUS, or huge IPs. User can find resets of this type in CMU register map, such as **STOP\_AND\_GO\_RESET\_SYS\_0\_AXI\_CLK**.

Next type is RSTCON reset. This reset controlled by RSTCON, not CMU. When user release this reset, for example set 1 to **RST\_sdmmc\_0\_axi\_rst**, there are two type of operation.

When user set 1 to **RSTMODE\_sdmmc\_0\_axi\_rst**, then sdmmc\_0\_axi\_rst is released directly. So, user must set disable sdmmc\_0\_axi\_clk before release sdmmc\_0\_axi\_rst for safe reset release. After release reset, user must set enable sdmmc\_0\_axi\_clk for using IP.

When user set 0 to **RSTMODE\_sdmmc\_0\_axi\_rst**, then sdmmc\_0\_axi\_rst is released synchronous at posedge of sdmmc\_0\_axi\_clk. So, user must sdmmc\_0\_axi\_clk clock enable before reset release.

User must set 1 to **RSTMODE\_\*\_rst** while using DRONE\_SOC.

## 6.2 Features

- Supports mode for reset release without sync-clock.
- Supports glitch-less mux for bus clock divider.

## 6.3 Application Guide

### 6.3.1 S/W Reset Release Guide

When user want to release resets of IP, user must release resets of IP when clocks of IP are disabled. This sequence must be adapted all IP except some resets of IPs are released automatic after power-on. Followings are sequence for release reset.

works of stop and go reset/reset mode register is described in Overview. Please see Overview section.

1. Set 0 to clock enable registers of IP.
2. Set 1 to reset mode control registers of IP.
3. Set 1 to reset control registers of IP. (reset release)
4. Set 1 to clock enable registers of IP.

## 6.4 Clock Source Description

Following table shows simple description about clock source.

Clock Source Name	Description
PLL0_CLK	PLL[0] clock for System
PLL1_CLK	PLL[1] clock for RISC-V

**Table 6.1:** Clock Sources

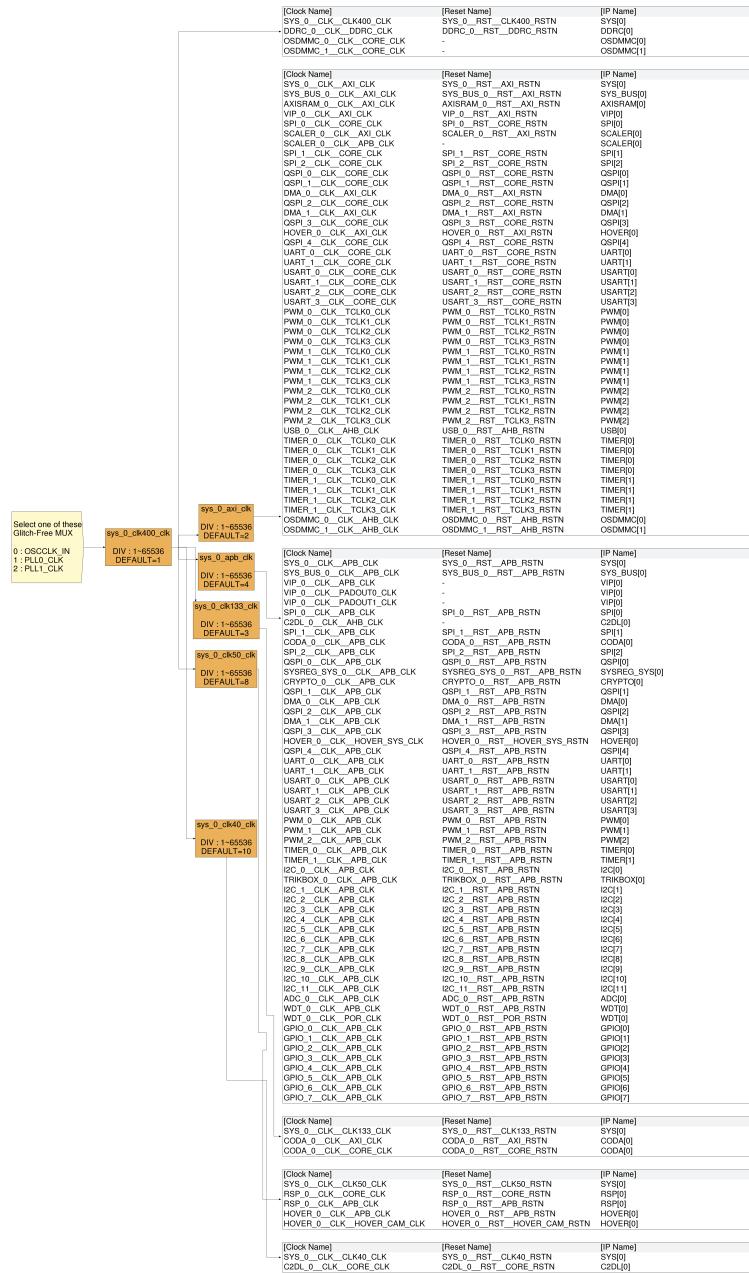
## 6.5 CMU\_SYS

Clock group list of CMU\_SYS

- SYS\_0\_CLK400\_CLK(With PLL0)
- CPU\_0\_CORE\_CLK(With PLL1)

## 6.5.1 SYS\_0\_CLK400\_CLK

### 6.5.1.1 Block Diagram



### 6.5.1.2 Register Map Summary

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)

Register	Offset	Description	Reset Value
CLKMUX_SYS_0_CLK400_CLK	0x00	Clock MUX Select for SYS_0_CLK400_CLK	0x0000_0000
GRPRST_SYS_0_CLK400_CLK	0x04	Group Reset for SYS_0_CLK400_CLK	0x0000_0001
GATEMODE_SYS_0_CLK400_CLK	0x08	Gate Mode for SYS_0_CLK400_CLK	0x0000_0001
CLKENB_SYS_0_CLK400_CLK0	0x0C	Clock Enable for SYS_0_CLK400_CLK	0xFFFF_FFFF
CLKENB_SYS_0_CLK400_CLK1	0x10	Clock Enable for SYS_0_CLK400_CLK	0xFFFF_FFFF
CLKENB_SYS_0_CLK400_CLK2	0x14	Clock Enable for SYS_0_CLK400_CLK	0xFFFF_FFFF
CLKENB_SYS_0_CLK400_CLK3	0x18	Clock Enable for SYS_0_CLK400_CLK	0x0007_FFFF
DIVRST_SYS_0_CLK400_CLK	0x44	Divider Reset for SYS_0_CLK400_CLK	0x0000_0001
DIVVAL_SYS_0_CLK400_CLK	0x48	Divider Reset for SYS_0_CLK400_CLK	0x0000_0001
STOP_AND_GO_RESET_SYS_0_CLK400_CLK	0x4C	Stop-and-go Reset Control for SYS_0_CLK400_CLK	0x0000_0001
DIVSTATUS_SYS_0_CLK400_CLK	0x50	Divider Status for SYS_0_CLK400_CLK	0x0000_0000
DIVRST_SYS_0_AXI_CLK	0x84	Divider Reset for SYS_0_AXI_CLK	0x0000_0001
DIVVAL_SYS_0_AXI_CLK	0x88	Divider Reset for SYS_0_AXI_CLK	0x0000_0001
STOP_AND_GO_RESET_SYS_0_AXI_CLK	0x8C	Stop-and-go Reset Control for SYS_0_AXI_CLK	0x0000_0001
DIVSTATUS_SYS_0_AXI_CLK	0x90	Divider Status for SYS_0_AXI_CLK	0x0000_0000
DIVRST_SYS_0_APB_CLK	0xC4	Divider Reset for SYS_0_APB_CLK	0x0000_0001
DIVVAL_SYS_0_APB_CLK	0xC8	Divider Reset for SYS_0_APB_CLK	0x0000_0001
STOP_AND_GO_RESET_SYS_0_APB_CLK	0xCC	Stop-and-go Reset Control for SYS_0_APB_CLK	0x0000_0001
DIVSTATUS_SYS_0_APB_CLK	0xD0	Divider Status for SYS_0_APB_CLK	0x0000_0000
DIVRST_SYS_0_CLK133_CLK	0x104	Divider Reset for SYS_0_CLK133_CLK	0x0000_0001
DIVVAL_SYS_0_CLK133_CLK	0x108	Divider Reset for SYS_0_CLK133_CLK	0x0000_0001
STOP_AND_GO_RESET_SYS_0_CLK133_CLK	0x10C	Stop-and-go Reset Control for SYS_0_CLK133_CLK	0x0000_0001
DIVSTATUS_SYS_0_CLK133_CLK	0x110	Divider Status for SYS_0_CLK133_CLK	0x0000_0000
DIVRST_SYS_0_CLK50_CLK	0x144	Divider Reset for SYS_0_CLK50_CLK	0x0000_0001
DIVVAL_SYS_0_CLK50_CLK	0x148	Divider Reset for SYS_0_CLK50_CLK	0x0000_0001
STOP_AND_GO_RESET_SYS_0_CLK50_CLK	0x14C	Stop-and-go Reset Control for SYS_0_CLK50_CLK	0x0000_0001
DIVSTATUS_SYS_0_CLK50_CLK	0x150	Divider Status for SYS_0_CLK50_CLK	0x0000_0000
DIVRST_SYS_0_CLK40_CLK	0x184	Divider Reset for SYS_0_CLK40_CLK	0x0000_0001
DIVVAL_SYS_0_CLK40_CLK	0x188	Divider Reset for SYS_0_CLK40_CLK	0x0000_0001
STOP_AND_GO_RESET_SYS_0_CLK40_CLK	0x18C	Stop-and-go Reset Control for SYS_0_CLK40_CLK	0x0000_0001

DIVSTATUS_SYS_0_CLK40_CLK	0x190	Divider Status for SYS_0_CLK40_CLK	0x0000_0000
---------------------------	-------	------------------------------------	-------------

### 6.5.1.3 CLKMUX\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
CLKMUX_SYS_0_CLK400_CLK	[1:0]	R/W	0 : OSCCLK_IN 1 : PLL_CLK 2 : PLL1_CLK	0

### 6.5.1.4 GRPRST\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
GRPRST_SYS_0_CLK400_CLK	[0]	R/W	for special use only	1

### 6.5.1.5 GATEMODE\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
GATEMODE_SYS_0_CLK400_CLK	[1:0]	R/W	0: ALL_OFF 1: SEPERATE CONTROL 2: ALL_ON	1

### 6.5.1.6 CLKENB\_SYS\_0\_CLK400\_CLK0

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x0C Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
CLKENB_PWM_1_CLK_TCLK0_CLK	[31]	R/W	enable PWM_1_CLK_TCLK0_CLK 0: disable, 1: enable	1
CLKENB_PWM_0_CLK_TCLK3_CLK	[30]	R/W	enable PWM_0_CLK_TCLK3_CLK 0: disable, 1: enable	1
CLKENB_PWM_0_CLK_TCLK2_CLK	[29]	R/W	enable PWM_0_CLK_TCLK2_CLK 0: disable, 1: enable	1
CLKENB_PWM_0_CLK_TCLK1_CLK	[28]	R/W	enable PWM_0_CLK_TCLK1_CLK 0: disable, 1: enable	1
CLKENB_PWM_0_CLK_TCLK0_CLK	[27]	R/W	enable PWM_0_CLK_TCLK0_CLK 0: disable, 1: enable	1

CLKENB_USART_3_CLK_CORE_CLK	[26]	R/W	enable USART_3_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_USART_2_CLK_CORE_CLK	[25]	R/W	enable USART_2_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_USART_1_CLK_CORE_CLK	[24]	R/W	enable USART_1_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_USART_0_CLK_CORE_CLK	[23]	R/W	enable USART_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_UART_1_CLK_CORE_CLK	[22]	R/W	enable UART_1_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_UART_0_CLK_CORE_CLK	[21]	R/W	enable UART_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_QSPI_4_CLK_CORE_CLK	[20]	R/W	enable QSPI_4_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_HOVER_0_CLK_AXI_CLK	[19]	R/W	enable HOVER_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_QSPI_3_CLK_CORE_CLK	[18]	R/W	enable QSPI_3_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_DMA_1_CLK_AXI_CLK	[17]	R/W	enable DMA_1_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_QSPI_2_CLK_CORE_CLK	[16]	R/W	enable QSPI_2_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_DMA_0_CLK_AXI_CLK	[15]	R/W	enable DMA_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_QSPI_1_CLK_CORE_CLK	[14]	R/W	enable QSPI_1_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_QSPI_0_CLK_CORE_CLK	[13]	R/W	enable QSPI_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_SPI_2_CLK_CORE_CLK	[12]	R/W	enable SPI_2_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_SPI_1_CLK_CORE_CLK	[11]	R/W	enable SPI_1_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_SCALER_0_CLK_APB_CLK	[10]	R/W	enable SCALER_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_SCALER_0_CLK_AXI_CLK	[9]	R/W	enable SCALER_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_SPI_0_CLK_CORE_CLK	[8]	R/W	enable SPI_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_VIP_0_CLK_AXI_CLK	[7]	R/W	enable VIP_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_AXISRAM_0_CLK_AXI_CLK	[6]	R/W	enable AXISRAM_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_SYS_BUS_0_CLK_AXI_CLK	[5]	R/W	enable SYS_BUS_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_SYS_0_CLK_AXI_CLK	[4]	R/W	enable SYS_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_OSDMMC_1_CLK_CORE_CLK	[3]	R/W	enable OSDMMC_1_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_OSDMMC_0_CLK_CORE_CLK	[2]	R/W	enable OSDMMC_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_DDRC_0_CLK_DDRC_CLK	[1]	R/W	enable DDRC_0_CLK_DDRC_CLK 0: disable, 1: enable	1
CLKENB_SYS_0_CLK_CLK400_CLK	[0]	R/W	enable SYS_0_CLK_CLK400_CLK 0: disable, 1: enable	1

#### 6.5.1.7 CLKENB\_SYS\_0\_CLK400\_CLK1

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x10 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

CLKENB_QSPI_1__CLK_APB_CLK	[31]	R/W	enable QSPI_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_CRYPTO_0__CLK_APB_CLK	[30]	R/W	enable CRYPTO_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_SYSREG_SYS_0_CLK_APB_CLK	[29]	R/W	enable SYSREG_SYS_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_QSPI_0__CLK_APB_CLK	[28]	R/W	enable QSPI_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_SPI_2__CLK_APB_CLK	[27]	R/W	enable SPI_2_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_CODA_0__CLK_APB_CLK	[26]	R/W	enable CODA_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_SPI_1__CLK_APB_CLK	[25]	R/W	enable SPI_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_C2DL_0__CLK_AHB_CLK	[24]	R/W	enable C2DL_0_CLK_AHB_CLK 0: disable, 1: enable	1
CLKENB_SPL_0__CLK_APB_CLK	[23]	R/W	enable SPL_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_VIP_0__CLK_PADOUT1_CLK	[22]	R/W	enable VIP_0_CLK_PADOUT1_CLK 0: disable, 1: enable	1
CLKENB_VIP_0__CLK_PADOUT0_CLK	[21]	R/W	enable VIP_0_CLK_PADOUT0_CLK 0: disable, 1: enable	1
CLKENB_VIP_0__CLK_APB_CLK	[20]	R/W	enable VIP_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_SYS_BUS_0__CLK_APB_CLK	[19]	R/W	enable SYS_BUS_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_SYS_0__CLK_APB_CLK	[18]	R/W	enable SYS_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_OSDMMC_1__CLK_AHB_CLK	[17]	R/W	enable OSDMMC_1_CLK_AHB_CLK 0: disable, 1: enable	1
CLKENB_OSDMMC_0__CLK_AHB_CLK	[16]	R/W	enable OSDMMC_0_CLK_AHB_CLK 0: disable, 1: enable	1
CLKENB_TIMER_1__CLK_TCLK3_CLK	[15]	R/W	enable TIMER_1_CLK_TCLK3_CLK 0: disable, 1: enable	1
CLKENB_TIMER_1__CLK_TCLK2_CLK	[14]	R/W	enable TIMER_1_CLK_TCLK2_CLK 0: disable, 1: enable	1
CLKENB_TIMER_1__CLK_TCLK1_CLK	[13]	R/W	enable TIMER_1_CLK_TCLK1_CLK 0: disable, 1: enable	1
CLKENB_TIMER_1__CLK_TCLK0_CLK	[12]	R/W	enable TIMER_1_CLK_TCLK0_CLK 0: disable, 1: enable	1
CLKENB_TIMER_0__CLK_TCLK3_CLK	[11]	R/W	enable TIMER_0_CLK_TCLK3_CLK 0: disable, 1: enable	1
CLKENB_TIMER_0__CLK_TCLK2_CLK	[10]	R/W	enable TIMER_0_CLK_TCLK2_CLK 0: disable, 1: enable	1
CLKENB_TIMER_0__CLK_TCLK1_CLK	[9]	R/W	enable TIMER_0_CLK_TCLK1_CLK 0: disable, 1: enable	1
CLKENB_TIMER_0__CLK_TCLK0_CLK	[8]	R/W	enable TIMER_0_CLK_TCLK0_CLK 0: disable, 1: enable	1
CLKENB_USB_0__CLK_AHB_CLK	[7]	R/W	enable USB_0_CLK_AHB_CLK 0: disable, 1: enable	1
CLKENB_PWM_2__CLK_TCLK3_CLK	[6]	R/W	enable PWM_2_CLK_TCLK3_CLK 0: disable, 1: enable	1
CLKENB_PWM_2__CLK_TCLK2_CLK	[5]	R/W	enable PWM_2_CLK_TCLK2_CLK 0: disable, 1: enable	1
CLKENB_PWM_2__CLK_TCLK1_CLK	[4]	R/W	enable PWM_2_CLK_TCLK1_CLK 0: disable, 1: enable	1
CLKENB_PWM_2__CLK_TCLK0_CLK	[3]	R/W	enable PWM_2_CLK_TCLK0_CLK 0: disable, 1: enable	1
CLKENB_PWM_1__CLK_TCLK3_CLK	[2]	R/W	enable PWM_1_CLK_TCLK3_CLK 0: disable, 1: enable	1
CLKENB_PWM_1__CLK_TCLK2_CLK	[1]	R/W	enable PWM_1_CLK_TCLK2_CLK 0: disable, 1: enable	1
CLKENB_PWM_1__CLK_TCLK1_CLK	[0]	R/W	enable PWM_1_CLK_TCLK1_CLK 0: disable, 1: enable	1

### 6.5.1.8 CLKENB\_SYS\_0\_CLK400\_CLK2

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x14 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
CLKENB_WDT_0_CLK_APB_CLK	[31]	R/W	enable WDT_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_ADC_0_CLK_APB_CLK	[30]	R/W	enable ADC_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_11_CLK_APB_CLK	[29]	R/W	enable I2C_11_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_10_CLK_APB_CLK	[28]	R/W	enable I2C_10_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_9_CLK_APB_CLK	[27]	R/W	enable I2C_9_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_8_CLK_APB_CLK	[26]	R/W	enable I2C_8_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_7_CLK_APB_CLK	[25]	R/W	enable I2C_7_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_6_CLK_APB_CLK	[24]	R/W	enable I2C_6_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_5_CLK_APB_CLK	[23]	R/W	enable I2C_5_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_4_CLK_APB_CLK	[22]	R/W	enable I2C_4_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_3_CLK_APB_CLK	[21]	R/W	enable I2C_3_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_2_CLK_APB_CLK	[20]	R/W	enable I2C_2_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_1_CLK_APB_CLK	[19]	R/W	enable I2C_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_TRIKBOX_0_CLK_APB_CLK	[18]	R/W	enable TRIKBOX_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_I2C_0_CLK_APB_CLK	[17]	R/W	enable I2C_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_TIMER_1_CLK_APB_CLK	[16]	R/W	enable TIMER_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_TIMER_0_CLK_APB_CLK	[15]	R/W	enable TIMER_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_PWM_2_CLK_APB_CLK	[14]	R/W	enable PWM_2_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_PWM_1_CLK_APB_CLK	[13]	R/W	enable PWM_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_PWM_0_CLK_APB_CLK	[12]	R/W	enable PWM_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_USART_3_CLK_APB_CLK	[11]	R/W	enable USART_3_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_USART_2_CLK_APB_CLK	[10]	R/W	enable USART_2_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_USART_1_CLK_APB_CLK	[9]	R/W	enable USART_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_USART_0_CLK_APB_CLK	[8]	R/W	enable USART_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_UART_1_CLK_APB_CLK	[7]	R/W	enable UART_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_UART_0_CLK_APB_CLK	[6]	R/W	enable UART_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_QSPI_4_CLK_APB_CLK	[5]	R/W	enable QSPI_4_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_HOVER_0_CLK_HOVER_SYS_CLK	[4]	R/W	enable HOVER_0_CLK_HOVER_SYS_CLK 0: disable, 1: enable	1
CLKENB_QSPI_3_CLK_APB_CLK	[3]	R/W	enable QSPI_3_CLK_APB_CLK 0: disable, 1: enable	1

CLKENB_DMA_1_CLK_APB_CLK	[2]	R/W	enable DMA_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_QSPI_2_CLK_APB_CLK	[1]	R/W	enable QSPI_2_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_DMA_0_CLK_APB_CLK	[0]	R/W	enable DMA_0_CLK_APB_CLK 0: disable, 1: enable	1

### 6.5.1.9 CLKENB\_SYS\_0\_CLK400\_CLK3

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x18 Reset Value = 0x0007\_FFFF

Name	Bit	Type	Description	Reset Value
RSVD	[31:22]	-	Reserved	-
CLKENB_C2DL_0_CLK_CORE_CLK	[18]	R/W	enable C2DL_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_SYS_0_CLK_CLK40_CLK	[17]	R/W	enable SYS_0_CLK_CLK40_CLK 0: disable, 1: enable	1
CLKENB_HOVER_0_CLK_HOVER_CAM_CLK	[16]	R/W	enable HOVER_0_CLK_HOVER_CAM_CLK 0: disable, 1: enable	1
CLKENB_HOVER_0_CLK_APB_CLK	[15]	R/W	enable HOVER_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_RSP_0_CLK_APB_CLK	[14]	R/W	enable RSP_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_RSP_0_CLK_CORE_CLK	[13]	R/W	enable RSP_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_SYS_0_CLK_CLK50_CLK	[12]	R/W	enable SYS_0_CLK_CLK50_CLK 0: disable, 1: enable	1
CLKENB_CODA_0_CLK_CORE_CLK	[11]	R/W	enable CODA_0_CLK_CORE_CLK 0: disable, 1: enable	1
CLKENB_CODA_0_CLK_AXI_CLK	[10]	R/W	enable CODA_0_CLK_AXI_CLK 0: disable, 1: enable	1
CLKENB_SYS_0_CLK_CLK133_CLK	[9]	R/W	enable SYS_0_CLK_CLK133_CLK 0: disable, 1: enable	1
CLKENB_GPIO_7_CLK_APB_CLK	[8]	R/W	enable GPIO_7_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_6_CLK_APB_CLK	[7]	R/W	enable GPIO_6_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_5_CLK_APB_CLK	[6]	R/W	enable GPIO_5_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_4_CLK_APB_CLK	[5]	R/W	enable GPIO_4_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_3_CLK_APB_CLK	[4]	R/W	enable GPIO_3_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_2_CLK_APB_CLK	[3]	R/W	enable GPIO_2_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_1_CLK_APB_CLK	[2]	R/W	enable GPIO_1_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_GPIO_0_CLK_APB_CLK	[1]	R/W	enable GPIO_0_CLK_APB_CLK 0: disable, 1: enable	1
CLKENB_WDT_0_CLK_POR_CLK	[0]	R/W	enable WDT_0_CLK_POR_CLK 0: disable, 1: enable	1

### 6.5.1.10 DIVRST\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x44 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVRST_SYS_0_CLK400_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1

#### 6.5.1.11 DIVVAL\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x48 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
DIVVAL_SYS_0_CLK400_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1

#### 6.5.1.12 STOP\_AND\_GO\_RESET\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x4C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_SYS_0_CLK400_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

#### 6.5.1.13 DIVSTATUS\_SYS\_0\_CLK400\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_SYS_0_CLK400_CLK	[0]	R	Divider Status . When user set DIVVAL_SYS_0_CLK400_CLK, user must wait until DIVSTATUS_SYS_0_CLK400_CLK becomes to 0. 0: Idle , 1: Busy	0

#### 6.5.1.14 DIVRST\_SYS\_0\_AXI\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x84 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-

DIVRST_SYS_0_AXI_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1
----------------------	-----	-----	--	---

#### 6.5.1.15 DIVVAL\_SYS\_0\_AXI\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x88 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
DIVVAL_SYS_0_AXI_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1

#### 6.5.1.16 STOP\_AND\_GO\_RESET\_SYS\_0\_AXI\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x8C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_SYS_0_AXI_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

#### 6.5.1.17 DIVSTATUS\_SYS\_0\_AXI\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x90 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_SYS_0_AXI_CLK	[0]	R	Divider Status . When user set DIVVAL_SYS_0_AXI_CLK, user must wait until DIVSTATUS_SYS_0_AXI_CLK becomes to 0. 0: Idle , 1: Busy	0

#### 6.5.1.18 DIVRST\_SYS\_0\_APB\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0xC4 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVRST_SYS_0_APB_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1

### 6.5.1.19 DIVVAL\_SYS\_0\_APB\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0xC8 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
DIVVAL_SYS_0_APB_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1

### 6.5.1.20 STOP\_AND\_GO\_RESET\_SYS\_0\_APB\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0xCC Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_SYS_0_APB_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

### 6.5.1.21 DIVSTATUS\_SYS\_0\_APB\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0xD0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_SYS_0_APB_CLK	[0]	R	Divider Status . When user set DIVVAL_SYS_0_APB_CLK, user must wait until DIVSTATUS_SYS_0_APB_CLK becomes to 0. 0: Idle , 1: Busy	0

### 6.5.1.22 DIVRST\_SYS\_0\_CLK133\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x104 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVRST_SYS_0_CLK133_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1

### 6.5.1.23 DIVVAL\_SYS\_0\_CLK133\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)

- Address = Base Address + 0x108 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
DIVVAL_SYS_0_CLK133_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1

#### 6.5.1.24 STOP\_AND\_GO\_RESET\_SYS\_0\_CLK133\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x10C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_SYS_0_CLK133_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

#### 6.5.1.25 DIVSTATUS\_SYS\_0\_CLK133\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x110 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_SYS_0_CLK133_CLK	[0]	R	Divider Status . When user set DIVVAL_SYS_0_CLK133_CLK, user must wait until DIVSTATUS_SYS_0_CLK133_CLK becomes to 0. 0: Idle , 1: Busy	0

#### 6.5.1.26 DIVRST\_SYS\_0\_CLK50\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x144 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVRST_SYS_0_CLK50_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1

#### 6.5.1.27 DIVVAL\_SYS\_0\_CLK50\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x148 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
DIVVAL_SYS_0_CLK50_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1

#### 6.5.1.28 STOP\_AND\_GO\_RESET\_SYS\_0\_CLK50\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x14C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_SYS_0_CLK50_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

#### 6.5.1.29 DIVSTATUS\_SYS\_0\_CLK50\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x150 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_SYS_0_CLK50_CLK	[0]	R	Divider Status . When user set DIVVAL_SYS_0_CLK50_CLK, user must wait until DIVSTATUS_SYS_0_CLK50_CLK becomes to 0. 0: Idle , 1: Busy	0

#### 6.5.1.30 DIVRST\_SYS\_0\_CLK40\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x184 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVRST_SYS_0_CLK40_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1

#### 6.5.1.31 DIVVAL\_SYS\_0\_CLK40\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x188 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-

DIVVAL_SYS_0_CLK40_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1
------------------------	--------	-----	---	---

#### 6.5.1.32 STOP\_AND\_GO\_RESET\_SYS\_0\_CLK40\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x18C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_SYS_0_CLK40_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

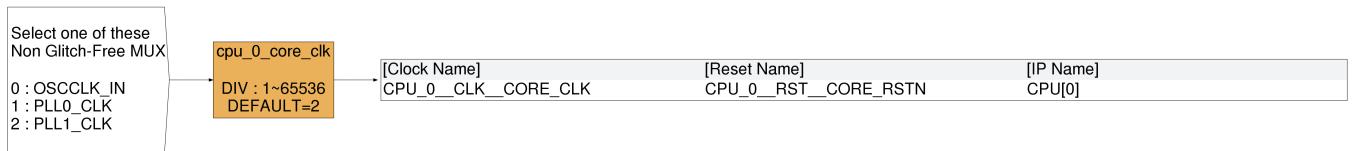
#### 6.5.1.33 DIVSTATUS\_SYS\_0\_CLK40\_CLK

- Base Address: 0x20010000+ 0x200(CMU\_SYS.SYS\_0\_CLK400\_CLK)
- Address = Base Address + 0x190 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_SYS_0_CLK40_CLK	[0]	R	Divider Status . When user set DIVVAL_SYS_0_CLK40_CLK, user must wait until DIVSTATUS_SYS_0_CLK40_CLK becomes to 0. 0: Idle , 1: Busy	0

## 6.5.2 CPU\_0\_CORE\_CLK

### 6.5.2.1 Block Diagram



### 6.5.2.2 Register Map Summary

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)

Register	Offset	Description	Reset Value
CLKMUX_CPU_0_CORE_CLK	0x00	Clock MUX Select for CPU_0_CORE_CLK	0x0000_0000
GRPRST_CPU_0_CORE_CLK	0x04	Group Reset for CPU_0_CORE_CLK	0x0000_0001
GATEMODE_CPU_0_CORE_CLK	0x08	Gate Mode for CPU_0_CORE_CLK	0x0000_0001
CLKENB_CPU_0_CORE_CLK0	0x0C	Clock Enable for CPU_0_CORE_CLK	0x0000_0001
DIVRST_CPU_0_CORE_CLK	0x44	Divider Reset for CPU_0_CORE_CLK	0x0000_0001
DIVVAL_CPU_0_CORE_CLK	0x48	Divider Reset for CPU_0_CORE_CLK	0x0000_0001
STOP_AND_GO_RESET_CPU_0_CORE_CLK	0x4C	Stop-and-go Reset Control for CPU_0_CORE_CLK	0x0000_0001
DIVSTATUS_CPU_0_CORE_CLK	0x50	Divider Status for CPU_0_CORE_CLK	0x0000_0000

### 6.5.2.3 CLKMUX\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
CLKMUX_CPU_0_CORE_CLK	[1:0]	R/W	0 : OSCCLK_IN 1 : PLL0_CLK 2 : PLL1_CLK	0

### 6.5.2.4 GRPRST\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
GRPRST_CPU_0_CORE_CLK	[0]	R/W	for special use only	1

### 6.5.2.5 GATEMODE\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
GATEMODE_CPU_0_CORE_CLK	[1:0]	R/W	0: ALL_OFF 1: SEPERATE CONTROL 2: ALL_ON	1

### 6.5.2.6 CLKENB\_CPU\_0\_CORE\_CLK0

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
CLKENB_CPU_0_CLK_CORE_CLK	[0]	R/W	enable CPU_0_CLK_CORE_CLK 0: disable, 1: enable	1

### 6.5.2.7 DIVRST\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x44 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVRST_CPU_0_CORE_CLK	[0]	R/W	0 : Divider Enter Reset, 1 : Divider Release Reset	1

#### 6.5.2.8 DIVVAL\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x48 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
DIVVAL_CPU_0_CORE_CLK	[15:0]	R/W	Divider Value 0 : Bypass, 1 - 131071 : regval + 1 Divider (odd value supports)	1

#### 6.5.2.9 STOP\_AND\_GO\_RESET\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x4C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
STOP_AND_GO_RESET_CPU_0_CORE_CLK	[0]	R/W	0: Enter Reset 1: Release Reset	1

#### 6.5.2.10 DIVSTATUS\_CPU\_0\_CORE\_CLK

- Base Address: 0x20010000+ 0x400(CMU\_SYS.CPU\_0\_CORE\_CLK)
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DIVSTATUS_CPU_0_CORE_CLK	[0]	R	Divider Status . When user set DIVVAL_CPU_0_CORE_CLK, user must wait until DIVSTATUS_CPU_0_CORE_CLK becomes to 0. 0: Idle , 1: Busy	0

# 7 USB2.0 DEVICE

## 7.1 Features

The USB2.0 DEVICE features include the following :

- The USB 2.0 Specification
- Operates in High-Speed (480 Mbps), Full-Speed (12 Mbps) modes
- Supports UTMI+ Level 3 interface
- 1 Control Endpoint 0 for control transfer
- 3 Device Mode Endpoints including Control Endpoint 0
- Programmable endpoint type: Bulk, Isochronous, or Interrupt
- Programmable IN/ OUT direction

## 7.2 Block Diagram

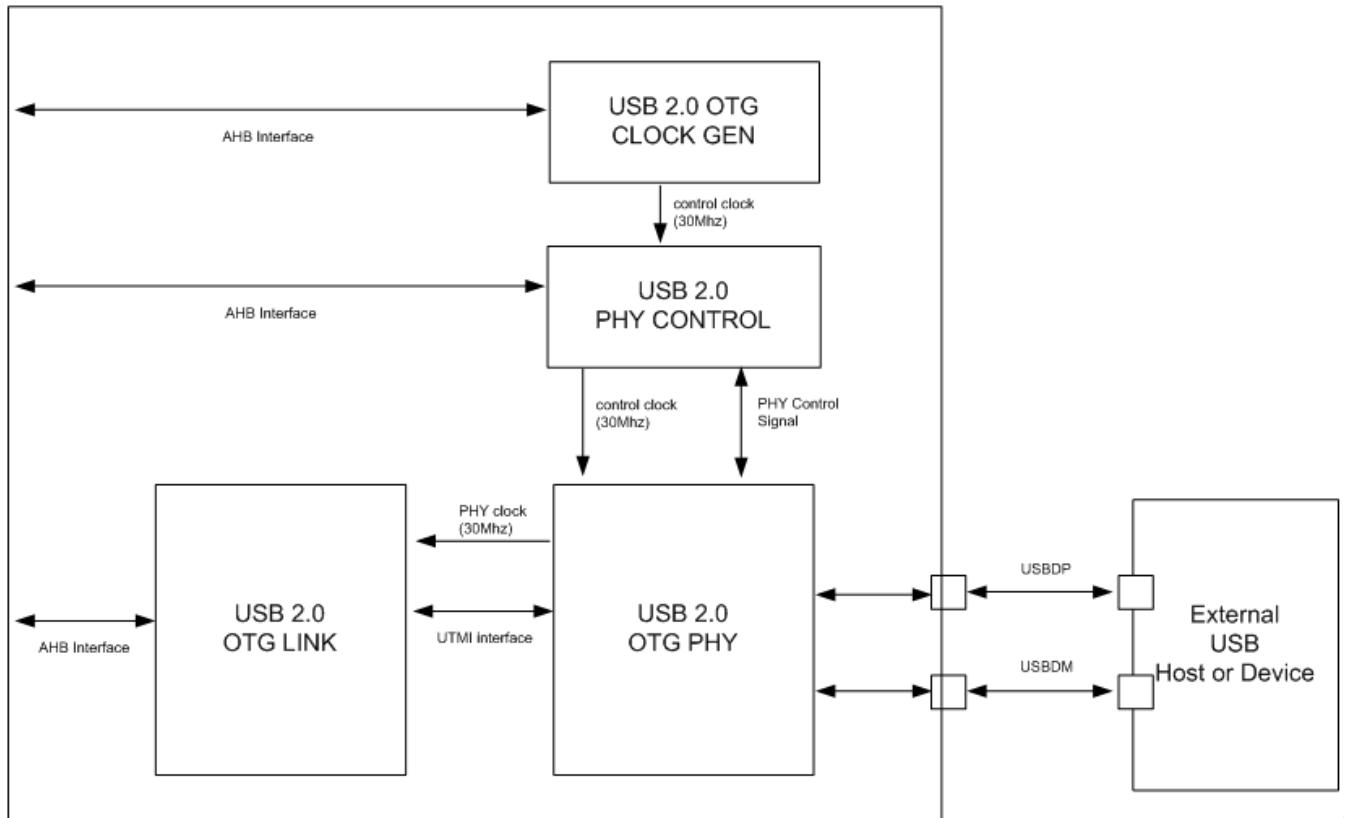


Figure 7.1: Block diagram of USB2.0 DEVICE

## 7.3 Functional Description

### 7.3.1 End Point Packet Size

USB2.0 DEVICE of this chip support 3 end-points. The maximum packet size per each EP(Endpoint) is as follows

EP number	Max Packet Size (bytes)
EP[0]	64
EP[1]	1024
EP[2]	1024

Table 7.1: maximum packet size per endpoint

### 7.3.2 Modes of Operation

The application operates the Link either in DMA mode or in Slave mode. The application cannot operate the core using DMA and Slave modes simultaneously

- DMA mode  
USB OTG host uses the AHB Master interface to transmit packet data fetch (AHB to USB) and receive data update (USB to AHB). The AHB master uses the programmed DMA address (DIEPDMA<sub>n</sub>/ DOEPDMA<sub>n</sub> register) to access the data buffers.
- Slave Mode  
USB OTG can operate either in transaction-level operation or in pipelined transaction-level operation. The application handles one data packet at a time per channel / endpoint in transaction-level operations. In pipelined transaction-level operation, the application programs the OTG to perform multiple transactions. The advantage of pipelined operation is that the application is not interrupted on packet basis

### 7.3.3 Programming User Config of PHY and LINK

- POR(Power On Reset) of PHY
  - program USB\_PHY\_CHG\_2[3:2](addr: 0x20040098) to 2'b10
- Wait clock of PHY : about 40 micro seconds
- Release utmi reset of LINK
  - program USB\_PHY\_CHG\_2[0](addr: 0x20040098) to 1'b1
- Release ahb reset of LINK
  - program USB\_PHY\_CHG\_3[0](addr: 0x20040128) to 1'b1

## 7.4 USB2.0 DEVICE Controller Register Description

### 7.4.1 Register Map Summary

- Base Address: 20D0\_0000(USB20OTG)

Register	Offset	Description	Reset Value
GOTGCTL	0x0	OTG Control and Status Register	0x032C0000
GOTGINT	0x4	OTG Interrupt Register	0x0
GAHBCFG	0x8	AHB Configuration Register	0x0
GUSBCFG	0xC	USB Configuration Register	0x1400
GRSTCTL	0x10	Reset Register	0x80000000
GINTSTS	0x14	Interrupt Register	0x14000020
GINTMSK	0x18	Interrupt Mask Register	0x0
GRXSTSR	0x1C	Receive Status Debug Read Register	0x0
GRXSTSP	0x20	Receive Status Read /Pop Register	0x0
GRXFPSZ	0x24	Receive FIFO Size Register	0x16CE
GNPTXFSIZ	0x28	Non-periodic Transmit FIFO Size Register	0x40016CE
GNPTXSTS	0x2C	Non-periodic Transmit FIFO/Queue Status Register	0x80400
GGPIO	0x38	General Purpose Input/Output Register	0x0
GUID	0x3C	User ID Register	0x12345678
GSNPSID	0x40	Synopsys ID Register	0x4F54330A
GHWCFG1	0x44	User HW Config1 Register	0x0
GHWCFG2	0x48	User HW Config2 Register	0x208FFC50
GHWCFG3	0x4C	User HW Config3 Register	0x164EF0E8
GHWCFG4	0x50	User HW Config4 Register	0xFFFF08030
GLPMCFG	0x54	LPM Config Register	0x0
GPWRDN	0x58	Global Power Down register	0x10
GDFIFOFCFG	0x5C	Global DFIFO Configuration Register	0x164E16CE
GADPCTL	0x60	ADP Timer, Control and Status Register	0x0
HPTXFSIZ	0x100	Host Periodic Transmit FIFO Size Register	0x16CE184E
DIEPTXF1	0x104	Device IN Endpoint Transmit FIFO Size Register 1	0x4001ACE
DIEPTXF2	0x108	Device IN Endpoint Transmit FIFO Size Register 2	0x4001ECE
DIEPTXF3	0x10C	Device IN Endpoint Transmit FIFO Size Register 3	0x40022CE
DIEPTXF4	0x110	Device IN Endpoint Transmit FIFO Size Register 4	0x40026CE
DIEPTXF5	0x114	Device IN Endpoint Transmit FIFO Size Register 5	0x4002ACE
DIEPTXF6	0x118	Device IN Endpoint Transmit FIFO Size Register 6	0x4002ECE
DIEPTXF7	0x11C	Device IN Endpoint Transmit FIFO Size Register 7	0x40032CE
DIEPTXF8	0x120	Device IN Endpoint Transmit FIFO Size Register 8	0x40036CE
DIEPTXF9	0x124	Device IN Endpoint Transmit FIFO Size Register 9	0x4003ACE
DIEPTXF10	0x128	Device IN Endpoint Transmit FIFO Size Register 10	0x4003ECE
DIEPTXF11	0x12C	Device IN Endpoint Transmit FIFO Size Register 11	0x40042CE
DIEPTXF12	0x130	Device IN Endpoint Transmit FIFO Size Register 12	0x40046CE
DIEPTXF13	0x134	Device IN Endpoint Transmit FIFO Size Register 13	0x4004ACE
DIEPTXF14	0x138	Device IN Endpoint Transmit FIFO Size Register 14	0x4004ECE
DIEPTXF15	0x13C	Device IN Endpoint Transmit FIFO Size Register 15	0x40052CE
DCFG	0x800	Device Configuration Register	0x8200000
DCTL	0x804	Device Control Register	0x2
DSTS	0x808	Device Status Register	0x2
DIEPMSK	0x810	Device IN Endpoint Common Interrupt Mask Register	0x0
DOEPMSK	0x814	Device OUT Endpoint Common Interrupt Mask Register	0x0
DAINT	0x818	Device All Endpoints Interrupt Register	0x0
DAINTMSK	0x81C	Device All Endpoints Interrupt Mask Register	0x0
DVBUSDIS	0x828	Device VBUS Discharge Time Register	0x17D7

DVBUSPULSE	0x82C	Device VBUS Pulsing Time Register	0x5B8
DTHRCTL	0x830	Device Threshold Control Register	0xC100020
DIEPEMPMSK	0x834	Device IN Endpoint FIFO Empty Interrupt Mask Register	0x0
DIEPCTL0	0x900	Device Control IN Endpoint 0 Control Register	0x8000
DIEPINT0	0x908	Device IN Endpoint 0 Interrupt Register	0x80
DIEPTSIZ0	0x910	Device IN Endpoint 0 Transfer Size Register	0x0
DIEPDMA0	0x914	Device IN Endpoint 0 DMA Address Register	0x0
DTXFSTS0	0x918	Device IN Endpoint Transmit FIFO Status Register 0	0x0
DIEPDMA0B	0x91C	Device IN Endpoint 16 Buffer Address Register	0x0
DIEPCTL1	0x920	Device Control IN Endpoint 1 Control Register	0x0
DIEPINT1	0x928	Device IN Endpoint 1 Interrupt Register	0x80
DIEPTSIZ1	0x930	Device IN Endpoint 1 Transfer Size Register	0x0
DIEPDMA1	0x934	Device IN Endpoint 1 DMA Address Register	0x0
DTXFSTS1	0x938	Device IN Endpoint Transmit FIFO Status Register 1	0x0
DIEPDMA1B	0x93C	Device IN Endpoint 1 Buffer Address Register	0x0
DIEPCTL2	0x940	Device Control IN Endpoint 2 Control Register	0x0
DIEPINT2	0x948	Device IN Endpoint 2 Interrupt Register	0x80
DIEPTSIZ2	0x950	Device IN Endpoint 2 Transfer Size Register	0x0
DIEPDMA2	0x954	Device IN Endpoint 2 DMA Address Register	0x0
DTXFSTS2	0x958	Device IN Endpoint Transmit FIFO Status Register 2	0x0
DIEPDMA2B	0x95C	Device IN Endpoint 2 Buffer Address Register	0x0
DIEPCTL3	0x960	Device Control IN Endpoint 3 Control Register	0x0
DIEPINT3	0x968	Device IN Endpoint 3 Interrupt Register	0x80
DIEPTSIZ3	0x970	Device IN Endpoint 3 Transfer Size Register	0x0
DIEPDMA3	0x974	Device IN Endpoint 3 DMA Address Register	0x0
DTXFSTS3	0x978	Device IN Endpoint Transmit FIFO Status Register 3	0x0
DIEPDMA3B	0x97C	Device IN Endpoint 3 Buffer Address Register	0x0
DIEPCTL4	0x980	Device Control IN Endpoint 4 Control Register	0x0
DIEPINT4	0x988	Device IN Endpoint 4 Interrupt Register	0x80
DIEPTSIZ4	0x990	Device IN Endpoint 4 Transfer Size Register	0x0
DIEPDMA4	0x994	Device IN Endpoint 4 DMA Address Register	0x0
DTXFSTS4	0x998	Device IN Endpoint Transmit FIFO Status Register 4	0x0
DIEPDMA4B	0x99C	Device IN Endpoint 4 Buffer Address Register	0x0
DIEPCTL5	0x9A0	Device Control IN Endpoint 5 Control Register	0x0
DIEPINT5	0x9A8	Device IN Endpoint 5 Interrupt Register	0x80
DIEPTSIZ5	0x9B0	Device IN Endpoint 5 Transfer Size Register	0x0
DIEPDMA5	0x9B4	Device IN Endpoint 5 DMA Address Register	0x0
DTXFSTS5	0x9B8	Device IN Endpoint Transmit FIFO Status Register 5	0x0
DIEPDMA5B	0x9BC	Device IN Endpoint 5 Buffer Address Register	0x0
DIEPCTL6	0x9C0	Device Control IN Endpoint 6 Control Register	0x0
DIEPINT6	0x9C8	Device IN Endpoint 6 Interrupt Register	0x80
DIEPTSIZ6	0x9D0	Device IN Endpoint 6 Transfer Size Register	0x0
DIEPDMA6	0x9D4	Device IN Endpoint 6 DMA Address Register	0x0
DTXFSTS6	0x9D8	Device IN Endpoint Transmit FIFO Status Register 6	0x0
DIEPDMA6B	0x9DC	Device IN Endpoint 6 Buffer Address Register	0x0
DIEPCTL7	0x9E0	Device Control IN Endpoint 7 Control Register	0x0
DIEPINT7	0x9E8	Device IN Endpoint 7 Interrupt Register	0x80
DIEPTSIZ7	0x9F0	Device IN Endpoint 7 Transfer Size Register	0x0
DIEPDMA7	0x9F4	Device IN Endpoint 7 DMA Address Register	0x0
DTXFSTS7	0x9F8	Device IN Endpoint Transmit FIFO Status Register 7	0x0
DIEPDMA7B	0x9FC	Device IN Endpoint 7 Buffer Address Register	0x0
DIEPCTL8	0xA00	Device Control IN Endpoint 8 Control Register	0x0
DIEPINT8	0xA08	Device IN Endpoint 8 Interrupt Register	0x80
DIEPTSIZ8	0xA10	Device IN Endpoint 8 Transfer Size Register	0x0
DIEPDMA8	0xA14	Device IN Endpoint 8 DMA Address Register	0x0
DTXFSTS8	0xA18	Device IN Endpoint Transmit FIFO Status Register 8	0x0

DIEPDMAB8	0xA1C	Device IN Endpoint 8 Buffer Address Register	0x0
DIEPCTL9	0xA20	Device Control IN Endpoint 9 Control Register	0x0
DIEPINT9	0xA28	Device IN Endpoint 9 Interrupt Register	0x80
DIEPTSIZ9	0xA30	Device IN Endpoint 9 Transfer Size Register	0x0
DIEPDMA9	0xA34	Device IN Endpoint 9 DMA Address Register	0x0
DTXFSTS9	0xA38	Device IN Endpoint Transmit FIFO Status Register 9	0x0
DIEPDMAB9	0xA3C	Device IN Endpoint 9 Buffer Address Register	0x0
DIEPCTL10	0xA40	Device Control IN Endpoint 10 Control Register	0x0
DIEPINT10	0xA48	Device IN Endpoint 10 Interrupt Register	0x80
DIEPTSIZ10	0xA50	Device IN Endpoint 10 Transfer Size Register	0x0
DIEPDMA10	0xA54	Device IN Endpoint 10 DMA Address Register	0x0
DTXFSTS10	0xA58	Device IN Endpoint Transmit FIFO Status Register 10	0x0
DIEPDMAB10	0xA5C	Device IN Endpoint 10 Buffer Address Register	0x0
DIEPCTL11	0xA60	Device Control IN Endpoint 11 Control Register	0x0
DIEPINT11	0xA68	Device IN Endpoint 11 Interrupt Register	0x80
DIEPTSIZ11	0xA70	Device IN Endpoint 11 Transfer Size Register	0x0
DIEPDMA11	0xA74	Device IN Endpoint 11 DMA Address Register	0x0
DTXFSTS11	0xA78	Device IN Endpoint Transmit FIFO Status Register 11	0x0
DIEPDMAB11	0xA7C	Device IN Endpoint 11 Buffer Address Register	0x0
DIEPCTL12	0xA80	Device Control IN Endpoint 12 Control Register	0x0
DIEPINT12	0xA88	Device IN Endpoint 12 Interrupt Register	0x80
DIEPTSIZ12	0xA90	Device IN Endpoint 12 Transfer Size Register	0x0
DIEPDMA12	0xA94	Device IN Endpoint 12 DMA Address Register	0x0
DTXFSTS12	0xA98	Device IN Endpoint Transmit FIFO Status Register 12	0x0
DIEPDMAB12	0xA9C	Device IN Endpoint 12 Buffer Address Register	0x0
DIEPCTL13	0xAA0	Device Control IN Endpoint 13 Control Register	0x0
DIEPINT13	0xAA8	Device IN Endpoint 13 Interrupt Register	0x80
DIEPTSIZ13	0xAB0	Device IN Endpoint 13 Transfer Size Register	0x0
DIEPDMA13	0xAB4	Device IN Endpoint 13 DMA Address Register	0x0
DTXFSTS13	0xAB8	Device IN Endpoint Transmit FIFO Status Register 13	0x0
DIEPDMAB13	0 ABC	Device IN Endpoint 13 Buffer Address Register	0x0
DIEPCTL14	0xAC0	Device Control IN Endpoint 14 Control Register	0x0
DIEPINT14	0xAC8	Device IN Endpoint 14 Interrupt Register	0x80
DIEPTSIZ14	0xAD0	Device IN Endpoint 14 Transfer Size Register	0x0
DIEPDMA14	0xAD4	Device IN Endpoint 14 DMA Address Register	0x0
DTXFSTS14	0xAD8	Device IN Endpoint Transmit FIFO Status Register 14	0x0
DIEPDMAB14	0ADC	Device IN Endpoint 14 Buffer Address Register	0x0
DIEPCTL15	0xAE0	Device Control IN Endpoint 15 Control Register	0x0
DIEPINT15	0xAE8	Device IN Endpoint 15 Interrupt Register	0x80
DIEPTSIZ15	0xAF0	Device IN Endpoint 15 Transfer Size Register	0x0
DIEPDMA15	0xAF4	Device IN Endpoint 15 DMA Address Register	0x0
DTXFSTS15	0xAF8	Device IN Endpoint Transmit FIFO Status Register 15	0x0
DIEPDMAB15	0 AFC	Device IN Endpoint 15 Buffer Address Register	0x0
DOEPCTL0	0xB00	Device Control OUT Endpoint 0 Control Register	0x8000
DOEPINT0	0xB08	Device OUT Endpoint 0 Interrupt Register	0x0
DOEPTSIZ0	0xB10	Device OUT Endpoint 0 Transfer Size Register	0x0
DOEPDMA0	0xB14	Device OUT Endpoint 0 DMA Address Register	0x0
DOEPDMAB0	0xB1C	Device OUT Endpoint 16 Buffer Address Register	0x0
DOEPCCTL1	0xB20	Device Control OUT Endpoint 1 Control Register	0x0
DOEPINT1	0xB28	Device OUT Endpoint 1 Interrupt Register	0x0
DOEPTSIZ1	0xB30	Device OUT Endpoint 1 Transfer Size Register	0x0
DOEPDMA1	0xB34	Device OUT Endpoint 1 DMA Address Register	0x0
DOEPDMAB1	0xB3C	Device OUT Endpoint 1 Buffer Address Register	0x0
DOEPCCTL2	0xB40	Device Control OUT Endpoint 2 Control Register	0x0
DOEPINT2	0xB48	Device OUT Endpoint 2 Interrupt Register	0x0
DOEPTSIZ2	0xB50	Device OUT Endpoint 2 Transfer Size Register	0x0

DOEPDMA2	0xB54	Device OUT Endpoint 2 DMA Address Register	0x0
DOEPDMAB2	0xB5C	Device OUT Endpoint 2 Buffer Address Register	0x0
DOEPCCTL3	0xB60	Device Control OUT Endpoint 3 Control Register	0x0
DOEPINT3	0xB68	Device OUT Endpoint 3 Interrupt Register	0x0
DOEPTSIZ3	0xB70	Device OUT Endpoint 3 Transfer Size Register	0x0
DOEPDMA3	0xB74	Device OUT Endpoint 3 DMA Address Register	0x0
DOEPDMAB3	0xB7C	Device OUT Endpoint 3 Buffer Address Register	0x0
DOEPCCTL4	0xB80	Device Control OUT Endpoint 4 Control Register	0x0
DOEPINT4	0xB88	Device OUT Endpoint 4 Interrupt Register	0x0
DOEPTSIZ4	0xB90	Device OUT Endpoint 4 Transfer Size Register	0x0
DOEPDMA4	0xB94	Device OUT Endpoint 4 DMA Address Register	0x0
DOEPDMAB4	0xB9C	Device OUT Endpoint 4 Buffer Address Register	0x0
DOEPCCTL5	0xBA0	Device Control OUT Endpoint 5 Control Register	0x0
DOEPINT5	0xBA8	Device OUT Endpoint 5 Interrupt Register	0x0
DOEPTSIZ5	0xBB0	Device OUT Endpoint 5 Transfer Size Register	0x0
DOEPDMA5	0xBB4	Device OUT Endpoint 5 DMA Address Register	0x0
DOEPDMAB5	0BBC	Device OUT Endpoint 5 Buffer Address Register	0x0
DOEPCCTL6	0xBC0	Device Control OUT Endpoint 6 Control Register	0x0
DOEPINT6	0xBC8	Device OUT Endpoint 6 Interrupt Register	0x0
DOEPTSIZ6	0xBD0	Device OUT Endpoint 6 Transfer Size Register	0x0
DOEPDMA6	0xBD4	Device OUT Endpoint 6 DMA Address Register	0x0
DOEPDMAB6	0BDC	Device OUT Endpoint 6 Buffer Address Register	0x0
DOEPCCTL7	0xBE0	Device Control OUT Endpoint 7 Control Register	0x0
DOEPINT7	0xBE8	Device OUT Endpoint 7 Interrupt Register	0x0
DOEPTSIZ7	0xBF0	Device OUT Endpoint 7 Transfer Size Register	0x0
DOEPDMA7	0xBF4	Device OUT Endpoint 7 DMA Address Register	0x0
DOEPDMAB7	0BFC	Device OUT Endpoint 7 Buffer Address Register	0x0
DOEPCCTL8	0xC00	Device Control OUT Endpoint 8 Control Register	0x0
DOEPINT8	0xC08	Device OUT Endpoint 8 Interrupt Register	0x0
DOEPTSIZ8	0xC10	Device OUT Endpoint 8 Transfer Size Register	0x0
DOEPDMA8	0xC14	Device OUT Endpoint 8 DMA Address Register	0x0
DOEPDMAB8	0xC1C	Device OUT Endpoint 8 Buffer Address Register	0x0
DOEPCCTL9	0xC20	Device Control OUT Endpoint 9 Control Register	0x0
DOEPINT9	0xC28	Device OUT Endpoint 9 Interrupt Register	0x0
DOEPTSIZ9	0xC30	Device OUT Endpoint 9 Transfer Size Register	0x0
DOEPDMA9	0xC34	Device OUT Endpoint 9 DMA Address Register	0x0
DOEPDMAB9	0xC3C	Device OUT Endpoint 9 Buffer Address Register	0x0
DOEPCCTL10	0xC40	Device Control OUT Endpoint 10 Control Register	0x0
DOEPINT10	0xC48	Device OUT Endpoint 10 Interrupt Register	0x0
DOEPTSIZ10	0xC50	Device OUT Endpoint 10 Transfer Size Register	0x0
DOEPDMA10	0xC54	Device OUT Endpoint 10 DMA Address Register	0x0
DOEPDMAB10	0xC5C	Device OUT Endpoint 10 Buffer Address Register	0x0
DOEPCCTL11	0xC60	Device Control OUT Endpoint 11 Control Register	0x0
DOEPINT11	0xC68	Device OUT Endpoint 11 Interrupt Register	0x0
DOEPTSIZ11	0xC70	Device OUT Endpoint 11 Transfer Size Register	0x0
DOEPDMA11	0xC74	Device OUT Endpoint 11 DMA Address Register	0x0
DOEPDMAB11	0xC7C	Device OUT Endpoint 11 Buffer Address Register	0x0
DOEPCCTL12	0xC80	Device Control OUT Endpoint 12 Control Register	0x0
DOEPINT12	0xC88	Device OUT Endpoint 12 Interrupt Register	0x0
DOEPTSIZ12	0xC90	Device OUT Endpoint 12 Transfer Size Register	0x0
DOEPDMA12	0xC94	Device OUT Endpoint 12 DMA Address Register	0x0
DOEPDMAB12	0xC9C	Device OUT Endpoint 12 Buffer Address Register	0x0
DOEPCCTL13	0xCA0	Device Control OUT Endpoint 13 Control Register	0x0
DOEPINT13	0xCA8	Device OUT Endpoint 13 Interrupt Register	0x0
DOEPTSIZ13	0xCB0	Device OUT Endpoint 13 Transfer Size Register	0x0
DOEPDMA13	0xCB4	Device OUT Endpoint 13 DMA Address Register	0x0

DOEPDMAB13	0xCBC	Device OUT Endpoint 13 Buffer Address Register	0x0
DOEPCTL14	0xCC0	Device Control OUT Endpoint 14 Control Register	0x0
DOEPINT14	0xCC8	Device OUT Endpoint 14 Interrupt Register	0x0
DOEPTSIZ14	0xCD0	Device OUT Endpoint 14 Transfer Size Register	0x0
DOEPDMA14	0xCD4	Device OUT Endpoint 14 DMA Address Register	0x0
DOEPDMAB14	0xCDC	Device OUT Endpoint 14 Buffer Address Register	0x0
DOEPCTL15	0xCE0	Device Control OUT Endpoint 15 Control Register	0x0
DOEPINT15	0xCE8	Device OUT Endpoint 15 Interrupt Register	0x0
DOEPTSIZ15	0xCF0	Device OUT Endpoint 15 Transfer Size Register	0x0
DOEPDMA15	0xCF4	Device OUT Endpoint 15 DMA Address Register	0x0
DOEPDMAB15	0xCFC	Device OUT Endpoint 15 Buffer Address Register	0x0

### 7.4.2 GOTGCTL

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x0 Reset Value = 0x032C0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:28]	-	RESERVED	-
ChirpEn	[27]	R/W	Mode : Device Only This bit when programmed to 1'b1 results in the core asserting chirp_on before sending an actual Chirp "K" signal on USB. This bit is present only if OTG_BC_SUPPORT = 1.If OTG_BC_SUPPORT!=1, this bit is a reserved bit.	0x0
MultValldBC	[26:22]	R	Multi Valued ID pin (MultValldBC) Battery Charger ACA inputs in the following order: Bit 26 - rid_float. Bit 25 - rid_gnd Bit 24 - rid_a Bit 23 - rid_b Bit 22 - rid_c	0xC
CurMod	[21]	R	Mode: Host and Device Current Mode of Operation (CurMod) Indicates the current mode. 1'b0: Device mode 1'b1: Host mode	0x1
OTGVer	[20]	R/W	OTG Version (OTGVer) Indicates the OTG revision. 1'b0: OTG Version 1.3. In this version the core supports Data line pulsing and VBus pulsing for SRP. 1'b1: OTG Version 2.0. In this version the core supports only Data line pulsing for SRP.	0x0
BSesVld	[19]	R	Mode: Device only B-Session Valid (BSesVld) Indicates the Device mode transceiver status. 1'b0: B-session is not valid. 1'b1: B-session is valid. In OTG mode, you can use this bit to determine if the device is connected or disconnected. Note: If you do not enable OTG features (such as SRP and HNP), the read reset value will be 1.The vbus assigns the values internally for non-SRP or non-HNP configurations.	0x1
ASesVld	[18]	R	Mode: Host only A-Session Valid (ASesVld) Indicates the Host mode transceiver status. 1'b0: A-session is not valid 1'b1: A-session is valid Note: If you do not enabled OTG features (such as SRP and HNP), the read reset value will be 1.The vbus assigns the values internally for non-SRP or non-HNP configurations.	0x1
DbncTime	[17]	R	Mode: Host only Long/Short Debounce Time (DbncTime) Indicates the debounce time of a detected connection. 1'b0: Long debounce time, used for physical connections (100 ms + 2.5 micro-sec) 1'b1: Short debounce time, used for soft connections (2.5 micro-sec)	0x0
ConIDsts	[16]	R	Mode: Host and Device Connector ID Status (ConIDsts) Indicates the connector ID status on a connect event. 1'b0: The DWC_otg core is in A-Device mode 1'b1: The DWC_otg core is in B-Device mode	0x0

DbnceFltrBypass	[15]	R/W	Valid only when OTG signals Debounce Filters are selected in configuration Bypass Debounce filters for bvalid, vbusvalid, sessend, idig signals when enabled. 1'b0: Disabled 1'b1: Enabled Default Value: 1'b0	0x0
EHEn	[12]	R/W	Mode: SRP Capable Host Embedded Host Enable (EHEn) 1'b0: Disable Embedded Host Mode 1'b1: Enable Embedded Host Mode	0x0
DevHNPEn	[11]	R/W	Mode: Device only Device HNP Enabled (DevHNPEn) The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host. 1'b0: HNP is not enabled in the application 1'b1: HNP is enabled in the application	0x0
HstSetHNPEn	[10]	R/W	Mode: Host only Host Set HNP Enable (HstSetHNPEn) The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device. 1'b0: Host Set HNP is not enabled 1'b1: Host Set HNP is enabled	0x0
HNPReq	[9]	R/W	Mode: Device only HNP Request (HNPReq) The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is SET. The core clears this bit when the HstNegSucStsChng bit is cleared. 1'b0: No HNP request 1'b1: HNP request	0x0
HstNegScs	[8]	R	Mode: Device only Host Negotiation Success (HstNegScs) The core sets this bit when host negotiation is successful. The core clears this bit when the HNP Request (HNPReq) bit in this register is SET. 1'b0: Host negotiation failure 1'b1: Host negotiation success	0x0
BvalidOvVal	[7]	R/W	B-Peripheral Session Valid OverrideValue (BvalidOvVal) This bit is used to set Override value for Bvalid signal when GOTGCTL.BvalidOvEn is set. 1'b0 : Bvalid value is 1'b0 when GOTGCTL.BvalidOvEn =1 1'b1 : Bvalid value is 1'b1 when GOTGCTL.BvalidOvEn =1	0x0
BvalidOvEn	[6]	R/W	B-Peripheral Session Valid Override Enable (BvalidOvEn) This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.BvalidOvVal. 1'b1 : Internally Bvalid received from the PHY is overridden with GOTGCTL.BvalidOvVal. 1'b0 : Override is disabled and bvalid signal from the respective PHY selected is used internally by the core.	0x0
AvalidOvVal	[5]	R/W	A-Peripheral Session Valid OverrideValue (AvalidOvVal) This bit is used to set Override value for Avalid signal when GOTGCTL.AvalidOvEn is set. 1'b0 : Avalid value is 1'b0 when GOTGCTL.AvalidOvEn =1 1'b1 : Avalid value is 1'b1 when GOTGCTL.AvalidOvEn =1	0x0
AvalidOvEn	[4]	R/W	A-Peripheral Session Valid Override Enable (AvalidOvEn) This bit is used to enable/disable the software to override the Avalid signal using the GOTGCTL.AvalidOvVal. 1'b1 : Internally Avalid received from the PHY is overridden with GOTGCTL.AvalidOvVal. 1'b0 : Override is disabled and avalid signal from the respective PHY selected is used internally by the core	0x0
VbvalidOvVal	[3]	R/W	VBUS Valid OverrideValue (VbvalidOvVal) This bit is used to set Override value for vbusvalid signal when GOTGCTL.VbvalidOvEn is set. 1'b0 : vbusvalid value is 1'b0 when GOTGCTL.VbvalidOvEn =1 1'b1 : vbusvalid value is 1'b1 when GOTGCTL.VbvalidOvEn =1	0x0

VbvalidOvEn	[2]	R/W	<p>VBUS Valid Override Enable (VbvalidOvEn)</p> <p>This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.VbvalidOvVal.</p> <p>1'b1 : Internally Bvalid received from the PHY is overridden with GOTGCTL.VbvalidOvVal.</p> <p>1'b0 : Override is disabled and bvalid signal from the respective PHY selected is used internally by the core</p>	0x0
SesReq	[1]	R/W	<p>Mode: Device only</p> <p>Session Request (SesReq)</p> <p>The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is SET. The core clears this bit when the HstNegSucStsChng bit is cleared. If you use the USB 1.1 Full-Speed Serial Transceiver interface to initiate the session request, the application must wait until the VBUS discharges to 0.2 V, after the B-Session Valid bit in this register (GOTGCTL.BSesVld) is cleared. This discharge time varies between different PHYs and can be obtained from the PHY vendor.</p> <p>1'b0: No session request 1'b1: Session request</p>	0x0
SesReqScs	[0]	R	<p>Mode: Device only</p> <p>Session Request Success (SesReqScs)</p> <p>The core sets this bit when a session request initiation is successful.</p> <p>1'b0: Session request failure 1'b1: Session request success</p>	0x0

### 7.4.3 GOTGINT

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:21]	-	RESERVED	-
MultVallpChng	[20]	R/W	This bit when set indicates that there is a change in the value of at least one ACA pin value.	0x0
DbnceDone	[19]	R/W	Mode: Host only Debounce Done (DbnceDone) The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is SET in the Core USB Configuration register (GUSBCFG.HNPCap or GUSBCFG.SRPCap, respectively). This bit can be set only by the core and the application should write 1 to clear it.	0x0
ADevTOUTChg	[18]	R/W	Mode:Host and Device A-Device Timeout Change (ADevTOUTChg) The core sets this bit to indicate that the A-device has timed out WHILE waiting FOR the B-device to connect. This bit can be set only by the core and the application should write 1 to clear it.	0x0
HstNegDet	[17]	R/W	Mode:Host and Device Host Negotiation Detected (HstNegDet) The core sets this bit when it detects a host negotiation request on the USB. This bit can be set only by the core and the application should write 1 to clear it.	0x0
RSVD	[16:10]	-	RESERVED	-
HstNegSucStsChng	[9]	R/W	Mode:Host and Device Host Negotiation Success Status Change (HstNegSucStsChng) The core sets this bit on the success or failure of a USB host negotiation request. The application must read the Host Negotiation Success bit of the OTG Control and Status register (GOTGCTL.HstNegScs) to check For success or failure. This bit can be set only by the core and the application should write 1 to clear it.	0x0
SesReqSucStsChng	[8]	R/W	Mode:Host and Device Session Request Success Status Change (SesReqSucStsChng) The core sets this bit on the success or failure of a session request. The application must read the Session Request Success bit in the OTG Control and Status register (GOTGCTL.SesReqScs) to check For success or failure. This bit can be set only by the core and the application should write 1 to clear it.	0x0
RSVD	[7:3]	-	RESERVED	-
SesEndDet	[2]	R/W	Mode:Host and Device Session End Detected (SesEndDet) The core sets this bit when the utmiotg_bvalid signal is deasserted. This bit can be set only by the core and the application should write 1 to clear it.	0x0
RSVD	[1:0]	-	RESERVED	-

#### 7.4.4 GAHBCFG

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:25]	-	RESERVED	-
InvDescEndianess	[24]	R/W	Invert Descriptor Endianess (InvDescEndianess) 1'b0: Descriptor Endianness is same as AHB Master Endianness 1'b1: Descriptor Endianness is Little Endian if AHB Master Endianness is Big Endian. Descriptor Endianness is Big Endian if AHB Master Endianness is Little Endian.	0x0
AHBSingle	[23]	R/W	AHB Single Support (AHBSingle) This bit when programmed supports Single transfers for the remaining data in a transfer when the DWC_otg core is operating in DMA mode. 1'b0: The remaining data in the transfer is sent using INCR burst size. 1'b1: The remaining data in the transfer is sent using Single burst size. Note: if this feature is enabled, the AHB RETRY and SPLIT transfers still have INCR burst type. Enable this feature when the AHB Slave connected to the DWC_otg core does not support INCR burst (and when Split, and Retry transactions are not being used in the bus).	0x0
NotiAllDmaWrit	[22]	R/W	Notify All Dma Write Transactions (NotiAllDmaWrit) This bit is programmed to enable the System DMA Done functionality for all the DMA write Transactions corresponding to the Channel/Endpoint. This bit is valid only when GAHBCFG.RemMemSupp is set to 1. GAHBCFG.NotiAllDmaWrit = 1 - HSOTG core asserts int_dma_req for all the DMA write transactions on the AHB interface along with int_dma_done, chep_last_transact and chep_number signal informations. The core waits for sys_dma_done signal for all the DMA write transactions in order to complete the transfer of a particular Channel/Endpoint. GAHBCFG.NotiAllDmaWrit = 0 - HSOTG core asserts int_dma_req signal only for the last transaction of DMA write transfer corresponding to a particular Channel/Endpoint. Similarly, the core waits for sys_dma_done signal only for that transaction of DMA write to complete the transfer of a particular Channel/Endpoint.	0x0

RemMemSupp	[21]	R/W	<p>Remote Memory Support (RemMemSupp)</p> <p>This bit is programmed to enable the functionality to wait for the system DMA</p> <p>Done Signal for the DMA Write Transfers.</p> <p>GAHBCFG.RemMemSupp=1</p> <ul style="list-style-type: none"> <li>- The int_dma_req output signal is asserted when HSOTG DMA starts</li> <li>write transfer to the external memory. When the core is done with the</li> <li>Transfers it asserts int_dma_done signal to flag the completion of DMA</li> <li>writes from HSOTG. The core then waits for sys_dma_done signal from</li> <li>the system to proceed further and complete the Data Transfer corresponding to a particular Channel/Endpoint.</li> </ul> <p>GAHBCFG.RemMemSupp=0</p> <ul style="list-style-type: none"> <li>- The int_dma_req and int_dma_done signals are not asserted and the core proceeds with the assertion of the XferComp interrupt as soon as the DMA write transfer is done at the HSOTG Core Boundary and it doesn't wait for the sys_dma_done signal to complete the DATA transfers</li> </ul>	0x0
PTxFEmpLvl	[8]	R/W	<p>Mode:Host only</p> <p>Periodic TxFIFO Empty Level (PTxFEmpLvl)</p> <p>Indicates when the Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.PTxFEmp) is triggered. This bit is used only in Slave mode.</p> <p>1'b0: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is half empty</p> <p>1'b1: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is completely empty</p>	0x0
NPTxFEmpLvl	[7]	R/W	<p>Mode:Host and device</p> <p>Non-Periodic TxFIFO Empty Level (NPTxFEmpLvl)</p> <p>This bit is used only in Slave mode.</p> <p>In host mode and with Shared FIFO with device mode, this bit indicates when the Non-Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.NPTxFEmp) is triggered.</p> <p>With dedicated FIFO in device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (DIEPINTn.TxFEmp) is triggered.</p> <p>Host mode and with Shared FIFO with device mode:-</p> <p>1'b0: GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is half empty</p> <p>1'b1: GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is completely empty</p> <p>Dedicated FIFO in device mode :-</p> <p>1'b0: DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is half empty</p> <p>1'b1: DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is completely empty</p>	0x0
RSVD	[6]	-	RESERVED	-
DMAEn	[5]	R/W	<p>Mode:Host and device</p> <p>DMA Enable (DMAEn)</p> <p>1'b0: Core operates in Slave mode</p> <p>1'b1: Core operates in a DMA mode</p> <p>This bit is always 0 when Slave-Only mode has been selected</p>	0x0

HBstLen	[4:1]	R/W	<p>Mode:Host and device Burst Length/Type (HBstLen) This field is used in both External and Internal DMA modes. In External DMA mode, these bits appear on dma_burst[3:0] ports, which can be used by an external wrapper to interface the External DMA Controller interface to Synopsys DW_ahb_dmac or ARM PrimeCell. External DMA Mode defines the DMA burst length in terms of 32-bit words: 4'b0000: 1 word 4'b0001: 4 words 4'b0010: 8 words 4'b0011: 16 words 4'b0100: 32 words 4'b0101: 64 words 4'b0110: 128 words 4'b0111: 256 words Others: Reserved Internal DMA Mode AHB Master burst type: 4'b0000 Single 4'b0001 INCR 4'b0011 INCR4 4'b0101 INCR8 4'b0111 INCR16 Others: Reserved</p>	0x0
GiblIntrMsk	[0]	R/W	<p>Mode:Host and device Global Interrupt Mask (GiblIntrMsk) The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core. 1'b0: Mask the interrupt assertion to the application. 1'b1: Unmask the interrupt assertion to the application.</p>	0x0

#### 7.4.5 GUSBCFG

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC Reset Value = 0x1400

Name	Bit	Type	Description	Reset Value
CorruptTxPkt	[31]	W	<p>Mode:Host and device Corrupt Tx packet This bit is for debug purposes only. Never Set this bit to 1. The application should always write 1'b0 to this bit.</p>	0x0
ForceDevMode	[30]	R/W	<p>Mode:Host and device Force Device Mode (ForceDevMode) Writing a 1 to this bit forces the core to device mode irrespective of utmiotg_iddig input pin. 1'b0 : Normal Mode. 1'b1 : Force Device Mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 micro sec is sufficient.</p>	0x0
ForceHstMode	[29]	R/W	<p>Mode:Host and device Force Host Mode (ForceHstMode) Writing a 1 to this bit forces the core to host mode irrespective of utmiotg_iddig input pin. 1'b0 : Normal Mode. 1'b1 : Force Host Mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 micro sec is sufficient.</p>	0x0

TxEndDelay	[28]	R/W	Mode: Device only Tx End Delay (TxEndDelay) Writing 1'b1 to this bit enables the core to follow the TxEndDelay timings as per UTMI+ specification 1.05 section 4.1.5 for opmode signal during remote wakeup. 1'b0 : Normal Mode. 1'b1 : Tx End delay.	0x0
IC_USBCap	[26]	R	IC_USB-Capable (IC_USBCap) The application uses this bit to control the DWC_otg core's IC_USB capabilities. 1'b0: IC_USB PHY Interface is not selected. 1'b1: IC_USB PHY Interface is selected. This bit is writable only if IC_USB is selected	0x0
TermSelDLPulse	[22]	R/W	Mode:Device only TermSel DLLine Pulsing Selection (TermSelDLPulse) This bit selects utmi_termselect to drive data line pulse during SRP. 1'b0: Data line pulsing using utmi_txvalid (Default). 1'b1: Data line pulsing using utmi_termsel.	0x0
PhyPwrClkSel	[15]	R/W	Mode:Host and Device PHY Low-Power Clock Select (PhyPwrClkSel) Selects either 480-MHz or 48-MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48-MHz clock to save power. 1'b0: 480-MHz Internal PLL clock 1'b1: 48-MHz External Clock In 480 MHz mode, the UTMI interface operates at either 60 or 30-MHz, depending upon whether 8- or 16-bit data width is selected. In 48-MHz mode, the UTMI interface operates at 48 MHz in FS mode and at either 48 or 6 MHz in LS mode (depending on the PHY vendor). This bit drives the utmi_fsls_low_power core output signal, and is valid only For UTMI+ PHYs.	0x0
RSVD	[14]	-	RESERVED	-
USBTrdTIm	[13:10]	R/W	Mode: Device only USB Turnaround Time (USBTrdTIm) Sets the turnaround time in PHY clocks. Specifies the response time For a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIFO (SPRAM). This must be programmed to 4'h5: When the MAC interface is 16-bit UTMI+ . 4'h9: When the MAC interface is 8-bit UTMI+ . Note: The values above are calculated For the minimum AHB frequency of 30 MHz. USB turnaround time is critical For certification where long cables and 5-Hubs are used, so If you need the AHB to run at less than 30 MHz, and If USB turnaround time is not critical, these bits can be programmed to a larger value.	0x5
HNPCap	[9]	R/W	Mode:Host and Device HNP-Capable (HNPCap) The application uses this bit to control the DWC_otg core's HNP capabilities. 1'b0: HNP capability is not enabled. 1'b1: HNP capability is enabled.	0x0
SRPCap	[8]	R/W	Mode:Host and Device SRP-Capable (SRPCap) The application uses this bit to control the DWC_otg core SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session. 1'b0: SRP capability is not enabled. 1'b1: SRP capability is enabled.	0x0

PHYSel	[6]	R	<p>Mode:Host and Device USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver Select (PHYSel)</p> <p>The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver.</p> <p>1'b0: USB 2.0 high-speed UTMI+ or ULPI PHY 1'b1: USB 1.1 full-speed serial transceiver</p> <p>If a USB 1.1 Full-Speed Serial Transceiver interface was not selected in, this bit is always 0, with Write Only access.</p> <p>If a high-speed PHY interface was not selected in, this bit is always 1, with Write Only access.</p> <p>If both interface types were selected (parameters have non-zero values), the application uses this bit to select which interface is active, and access is Read and Write.</p>	0x0
FSIntf	[5]	R	<p>Mode:Host and Device Full-Speed Serial Interface Select (FSIntf)</p> <p>The application uses this bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface.</p> <p>1'b0: 6-pin unidirectional full-speed serial interface 1'b1: 3-pin bidirectional full-speed serial interface</p> <p>If a USB 1.1 Full-Speed Serial Transceiver interface was not selected, this bit is always 0, with Write Only access. If a USB 1.1 FS interface was selected, Then the application can Set this bit to select between the 3- and 6-pin interfaces, and access is Read and Write.</p>	0x0
ULPI_UTMI_Sel	[4]	R	<p>Mode:Host and Device ULPI or UTMI+ Select (ULPI_UTMI_Sel)</p> <p>The application uses this bit to select either a UTMI+ interface or ULPI Interface.</p> <p>1'b0: UTMI+ Interface 1'b1: ULPI Interface</p> <p>This bit is writable only If UTMI+ and ULPI was specified For High-Speed PHY Interface(s).</p>	0x0
PHYIf	[3]	R/W	<p>Mode:Host and Device PHY Interface (PHYIf)</p> <p>The application uses this bit to configure the core to support a UTMI+ PHY with an 8- or 16-bit interface. When a ULPI PHY is chosen, this must be Set to 8-bit mode.</p> <p>1'b0: 8 bits 1'b1: 16 bits</p> <p>This bit is writable only If UTMI+ and ULPI were selected</p>	0x0
TOutCal	[2:0]	R/W	<p>Mode:Host and Device HS/FS Timeout Calibration (TOutCal)</p> <p>The number of PHY clocks that the application programs in this field is added to the high-speed/full-speed interpacket timeout duration in the core to account For any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the linestate condition can vary from one PHY to another.</p> <p>The USB standard timeout value For high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value For full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock are:</p> <p>High-speed operation: One 30-MHz PHY clock = 16 bit times One 60-MHz PHY clock = 8 bit times</p> <p>Full-speed operation: One 30-MHz PHY clock = 0.4 bit times One 60-MHz PHY clock = 0.2 bit times One 48-MHz PHY clock = 0.25 bit times</p>	0x0

### 7.4.6 GRSTCTL

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x10 Reset Value = 0x80000000

Name	Bit	Type	Description	Reset Value
AHBIdle	[31]	R	Mode:Host and Device AHB Master Idle (AHBIdle) Indicates that the AHB Master State Machine is in the IDLE condition.	0x1
DMAReq	[30]	R	Mode:Host and Device DMA Request Signal (DMAReq) Indicates that the DMA request is in progress. Used For debug.	0x0
RSVD	[29:11]	-	RESERVED	-
TxFNum	[10:6]	R/W	Mode:Host and Device TxFIFO Number (TxFNum) This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit. 5'h0: - Non-periodic TxFIFO flush in Host mode - Non-periodic TxFIFO flush in device mode when in shared FIFO operation - Tx FIFO 0 flush in device mode when in dedicated FIFO mode 5'h1: - Periodic TxFIFO flush in Host mode - Periodic TxFIFO 1 flush in Device mode when in shared FIFO operation - TXFIFO 1 flush in device mode when in dedicated FIFO mode 5'h2: - Periodic TxFIFO 2 flush in Device mode when in shared FIFO operation - TXFIFO 2 flush in device mode when in dedicated FIFO mode ... 5'hF: - Periodic TxFIFO 15 flush in Device mode when in shared FIFO operation - TXFIFO 15 flush in device mode when in dedicated FIFO mode 5'h10: Flush all the transmit FIFOs in device or host mode.	0x0
TxFFlsh	[5]	R/W	Mode:Host and Device TxFIFO Flush (TxFFlsh) This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction. The application must write this bit only after checking that the core is neither writing to the TxFIFO nor reading from the TxFIFO. Verify using these registers: ReadNAK Effective Interrupt ensures the core is not reading from the FIFO WriteGRSTCTL.AHBIdle ensures the core is not writing anything to the FIFO. Flushing is normally recommended when FIFOs are reconfigured or when switching between Shared FIFO and Dedicated Transmit FIFO operation. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.	0x0
RxFFlsh	[4]	R/W	Mode:Host and Device RxFIFO Flush (RxFFlsh) The application can flush the entire RxFIFO using this bit, but must first ensure that the core is not in the middle of a transaction. The application must only write to this bit after checking that the core is neither reading from the RxFIFO nor writing to the RxFIFO. The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.	0x0

FrmCntrRst	[2]	R/W	<p>Mode:Host only Host Frame Counter Reset (FrmCntrRst) The application writes this bit to reset the (micro)frame number counter inside the core. When the (micro)frame counter is reset, the subsequent SOF sent out by the core has a (micro)frame number of 0. When application writes 1 to the bit, it might not be able to read back the value as it will get cleared by the core in a few clock cycles.</p>	0x0
PIUFSSftRst	[1]	R/W	<p>Mode:Host and Device PIU FS Dedicated Controller Soft Reset (PIUFSSftRst) Resets the PIU FS Dedicated Controller. All module state machines in FS Dedicated Controller of PIU are reset to the IDLE state. Used to reset the FS Dedicated controller in PIU in case of any PHY Errors like Loss of activity or Babble Error resulting in the PHY remaining in RX state for more than one frame boundary</p>	0x0
CSftRst	[0]	R/W	<p>Mode:Host and Device Core Soft Reset (CSftRst) Resets the hclk and phy_clock domains as follows: Clears the interrupts and all the CSR registers except the following register bits:  <ul style="list-style-type: none"> <li>- PCGCCTL.RstPdwnModule</li> <li>- PCGCCTL.GateHclk</li> <li>- PCGCCTL.PwrClmp</li> <li>- PCGCCTL.StopPPhyLPwrClkSelClk</li> <li>- GUSBCFG.PhyLPwrClkSel</li> <li>- GUSBCFG.DDRSel</li> <li>- GUSBCFG.PHYSel</li> <li>- GUSBCFG.FSIntf</li> <li>- GUSBCFG.ULPI_UTMI_Sel</li> <li>- GUSBCFG.PHYIf</li> <li>- GUSBCFG.TxEndDelay</li> <li>- GUSBCFG.TermSelDLPulse</li> <li>- GUSBCFG.ULPIClkSusM</li> <li>- GUSBCFG.ULPIAutoRes</li> <li>- GUSBCFG.ULPIFsLs</li> <li>- GPIO</li> <li>- GPWRDN</li> <li>- GADPCTL</li> <li>- HCFG.FSLSPclkSel</li> <li>- DCFG.DevSpd</li> <li>- DCTL.SftDiscon</li> </ul> - All module state machines All module state machines (except the AHB Slave Unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed. Any transactions on the AHB Master are terminated as soon as possible, after gracefully completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. When Hibernation or ADP feature is enabled, the PMU module is not reset by the Core Soft Reset. The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 PHY clocks before doing any access to the PHY domain (synchronization delay). Software must also check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation. Typically software reset is used during software development and also when you dynamically change the PHY selection bits in the USB configuration registers listed above. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.</p>	0x0

### 7.4.7 GINTSTS

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x14 Reset Value = 0x14000020

Name	Bit	Type	Description	Reset Value
WkUpInt	[31]	R/W	<p>Mode:Host and Device</p> <p>Resume/Remote Wakeup Detected Interrupt (WkUpInt)</p> <p>Wakeup Interrupt during Suspend(L2) or LPM(L1) state.</p> <p>During Suspend(L2):</p> <ul style="list-style-type: none"> <li>- Device Mode - This interrupt is asserted only when Host Initiated Resume is detected on USB.</li> <li>- Host Mode - This interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB.</li> </ul> <p>For more information, see 'Partial Power-Down and Clock Gating Programming Model' in the Programming Guide.</p> <p>During LPM(L1):</p> <ul style="list-style-type: none"> <li>- Device Mode - This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB.</li> <li>- Host Mode - This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB.</li> </ul> <p>For more information, see 'LPM Entry and Exit Programming Model' in the Programming Guide.</p>	0x0
SessReqInt	[30]	R/W	<p>Mode:Host and Device</p> <p>Session Request/New Session Detected Interrupt (SessReqInt)</p> <p>In Host mode, this interrupt is asserted when a session request is detected from the device. In Host mode, this interrupt is asserted when a session request is detected from the device.</p> <p>In Device mode, this interrupt is asserted when the utmisrp_bvalid signal goes high.</p> <p>For more information on how to use this interrupt, see 'Partial Power-Down and Clock Gating Programming Model' in the Programming Guide.</p>	0x0
DisconnectInt	[29]	R/W	<p>Mode:Host only</p> <p>Disconnect Detected Interrupt (DisconnectInt)</p> <p>Asserted when a device disconnect is detected.</p>	0x0
ConIDStsChng	[28]	R/W	<p>Mode:Host and Device</p> <p>Connector ID Status Change (ConIDStsChng)</p> <p>The core sets this bit when there is a change in connector ID status.</p>	0x1
LPM_Int	[27]	R/W	<p>Device Mode - This interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.</p> <p>Host Mode - This interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (GLPMCFG.RetryCnt).</p> <p>This field is valid only if the Core LPM Configuration register's LPMCapable (LPMCap) field is set to 1</p>	0x0
PTxFEmp	[26]	R	<p>Mode:Host only</p> <p>Periodic Tx FIFO Empty (PTxFEmp)</p> <p>This interrupt is asserted when the Periodic Transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the Periodic Request Queue. The half or completely empty status is determined by the Periodic Tx FIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.PTxFEmpLvl).</p>	0x1

HChInt	[25]	R	<p>Mode:Host only Host Channels Interrupt (HChInt)</p> <p>The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in Host mode). The application must read the Host All Channels Interrupt (HAINT) register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding Host Channel-n Interrupt (HCINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the HCINTn register to clear this bit.</p>	0x0
PrtInt	[24]	R	<p>Mode:Host only Host Port Interrupt (PrtInt)</p> <p>The core sets this bit to indicate a change in port status of one of the DWC_otg core ports in Host mode. The application must read the Host Port Control and Status (HPRT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the Host Port Control and Status register to clear this bit.</p>	0x0
ResetDet	[23]	R/W	<p>Mode: Device only Reset detected Interrupt (ResetDet)</p> <p>In Device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in Suspend. In Host mode, this interrupt is not asserted.</p>	0x0
FetSusp	[22]	R/W	<p>Mode: Device only Data Fetch Suspended (FetSusp)</p> <p>This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of Tx FIFO space or Request Queue space. This interrupt is used by the application for an endpoint mismatch algorithm.</p> <p>For example, after detecting an endpoint mismatch, the application:</p> <ul style="list-style-type: none"> <li>Sets a Global non-periodic IN NAK handshake</li> <li>Disables IN endpoints</li> <li>Flushes the FIFO</li> </ul> <p>Determines the token sequence from the IN Token Sequence Learning Queue</p> <p>Re-enables the endpoints</p> <p>Clears the Global non-periodic IN NAK handshake</p> <p>If the Global non-periodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an 'IN token received when FIFO empty' interrupt. The OTG then sends the host a NAK response. To avoid this scenario, the application can check the GINTSTS.FetSusp interrupt, which ensures that the FIFO is full before clearing a Global NAK handshake.</p> <p>Alternatively, the application can mask the IN token received when FIFO empty interrupt when clearing a Global IN NAK handshake.</p>	0x0
incomplP	[21]	R/W	<p>Incomplete Periodic Transfer (incomplP)</p> <p>In Host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending which are scheduled for the current microframe.</p> <p>Incomplete Isochronous OUT Transfer (incomplISOOUT)</p> <p>In Device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.</p>	0x0
incomplSOIN	[20]	R/W	<p>Mode: Device only Incomplete Isochronous IN Transfer (incomplSOIN)</p> <p>The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.</p> <p>Note: This interrupt is not asserted in Scatter/Gather DMA mode.</p>	0x0

OEPInt	[19]	R	Mode: Device only OUT Endpoints Interrupt (OEPInt) The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and Then read the corresponding Device OUT Endpoint-n Interrupt (DOEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DOEPINTn register to clear this bit.	0x0
IEPInt	[18]	R	Mode: Device only IN Endpoints Interrupt (IEPInt) The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the IN endpoint on Device IN Endpoint-n Interrupt (DIEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DIEPINTn register to clear this bit.	0x0
EPMIs	[17]	R/W	Mode: Device only Endpoint Mismatch Interrupt (EPMIs) Note: This interrupt is valid only in shared FIFO operation. Indicates that an IN token has been received For a non-periodic endpoint, but the data For another endpoint is present in the top of the Non-periodic Transmit FIFO and the IN endpoint mismatch count programmed by the application has expired.	0x0
EOPF	[15]	R/W	Mode: Device only End of Periodic Frame Interrupt (EOPF) Indicates that the period specified in the Periodic Frame Interval field of the Device Configuration register (DCFG.PerFrInt) has been reached in the current microframe.	0x0
ISOOutDrop	[14]	R/W	Mode: Device only Isochronous OUT Packet Dropped Interrupt (ISOOutDrop) The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.	0x0
EnumDone	[13]	R/W	Mode: Device only Enumeration Done (EnumDone) The core sets this bit to indicate that speed enumeration is complete. The application must read the Device Status (DSTS) register to obtain the enumerated speed.	0x0
USBRst	[12]	R/W	Mode: Device only USB Reset (USBRst) The core sets this bit to indicate that a reset is detected on the USB.	0x0
USBSusp	[11]	R/W	Mode: Device only USB Suspend (USBSusp) The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the linestate signal For an extended period of time.	0x0
ErlySusp	[10]	R/W	Mode: Device only Early Suspend (ErlySusp) The core sets this bit to indicate that an Idle state has been detected on the USB For 3 ms.	0x0
GOUTNakEff	[7]	R	Mode: Device only Global OUT NAK Effective (GOUTNakEff) Indicates that the Set Global OUT NAK bit in the Device Control register (DCTL.SGOUTNak), Set by the application, has taken effect in the core. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register (DCTL.CGOUTNak).	0x0

GINNakEff	[6]	R	Mode: Device only Global IN Non-periodic NAK Effective (GINNakEff) Indicates that the Set Global Non-periodic IN NAK bit in the Device Control register (DCTL.SGNPInNak) set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit Set by the application. This bit can be cleared by clearing the Clear Global Non-periodic IN NAK bit in the Device Control register (DCTL.CGNPInNak). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.	0x0
NPTxFEmp	[5]	R	Mode: Host and Device Non-periodic TxFIFO Empty (NPTxFEmp) This interrupt is asserted when the Non-periodic TxFIFO is either half or completely empty, and there is space For at least one Entry to be written to the Non-periodic Transmit Request Queue. The half or completely empty status is determined by the Non-periodic TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl).	0x1
RxFLvl	[4]	R	Mode: Host and Device RxFIFO Non-Empty (RxFLvl) Indicates that there is at least one packet pending to be read from the RxFIFO.	0x0
Sof	[3]	R/W	Mode: Host and Device Start of (micro)Frame (Sof) In Host mode, the core sets this bit to indicate that an SOF (FS), micro-SOF (HS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt. In Device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current (micro)Frame number. This interrupt is seen only when the core is operating at either HS or FS. Note: This register may return 1'b1 if read immediately after power on reset. If the register bit reads 1'b1 immediately after power on re-set it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.	0x0
OTGInt	[2]	R	Mode: Host and Device OTG Interrupt (OTGInt) The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the GOTGINT register to clear this bit.	0x0
ModeMis	[1]	R/W	Mode: Host and Device Mode Mismatch Interrupt (ModeMis) The core sets this bit when the application is trying to access: A Host mode register, when the core is operating in Device mode A Device mode register, when the core is operating in Host mode The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core. This bit can be set only by the core and the application should write 1 to clear it	0x0
CurMod	[0]	R	Mode: Host and Device Current Mode of Operation (CurMod) Indicates the current mode. 1'b0: Device mode 1'b1: Host mode	0x0

### 7.4.8 GINTMSK

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x18 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
WkUpIntMsk	[31]	R/W	Mode: Host and Device Resume/Remote Wakeup Detected Interrupt Mask The WakeUp bit is used for LPM state wake up in a way similar to that of wake up in suspend state. (WkUpIntMsk)	0x0
SessReqIntMsk	[30]	R/W	Mode: Host and Device Session Request/New Session Detected Interrupt Mask (SessReqIntMsk)	0x0
DisconnectIntMsk	[29]	R/W	Mode: Host and Device Disconnect Detected Interrupt Mask (DisconnectIntMsk)	0x0
ConIDStsChngMsk	[28]	R/W	Mode: Host and Device Connector ID Status Change Mask (ConIDStsChngMsk)	0x0
LPM_IntMsk	[27]	R/W	Mode: Host and Device LPM Transaction Received Interrupt (LPM_Int) LPM Transaction received interrupt Mask	0x0
PTxFEmpMsk	[26]	R/W	Mode: Host only Periodic TxFIFO Empty Mask (PTxFEmpMsk)	0x0
HChIntMsk	[25]	R/W	Mode: Host only Host Channels Interrupt Mask (HChIntMsk)	0x0
PrtIntMsk	[24]	R/W	Mode: Host only Host Port Interrupt Mask (PrtIntMsk)	0x0
ResetDetMsk	[23]	R/W	Mode: Device only Reset detected Interrupt Mask (ResetDetMsk)	0x0
FetSuspMsk	[22]	R/W	Mode: Device only Data Fetch Suspended Mask (FetSuspMsk)	0x0
incomplPMsK	[21]	R/W	Mode: Host only Incomplete Periodic Transfer Mask (incomplPMsK) Mode: Device only Incomplete Isochronous OUT Transfer Interrupt Mask (incomp-ISOOUTMsk)	0x0
incomplSOINMsk	[20]	R/W	Mode: Device only Incomplete Isochronous IN Transfer Mask (incomplSOINMsk)	0x0
OEPIntMsk	[19]	R/W	Mode: Device only OUT Endpoints Interrupt Mask (OEPIntMsk)	0x0
IEPIntMsk	[18]	R/W	Mode: Device only IN Endpoints Interrupt Mask (IEPIntMsk)	0x0
EPMisMsk	[17]	R/W	Mode: Device only Endpoint Mismatch Interrupt Mask (EPMisMsk)	0x0
EOPFMsk	[15]	R/W	Mode: Device only End of Periodic Frame Interrupt Mask (EOPFMsk)	0x0
ISOOutDropMsk	[14]	R/W	Mode: Device only Isochronous OUT Packet Dropped Interrupt Mask (ISOOutDropMsk)	0x0
EnumDoneMsk	[13]	R/W	Mode: Device only Enumeration Done Mask (EnumDoneMsk)	0x0
USBRstMsk	[12]	R/W	Mode: Device only USB Reset Mask (USBRstMsk)	0x0
USBSuspMsk	[11]	R/W	Mode: Device only USB Suspend Mask (USBSuspMsk)	0x0
ErlySuspMsk	[10]	R/W	Mode: Device only Early Suspend Mask (ErlySuspMsk)	0x0
GOUTNakEffMsk	[7]	R/W	Mode: Device only Global OUT NAK Effective Mask (GOUTNakEffMsk)	0x0
GINNakEffMsk	[6]	R/W	Mode: Device only Global Non-periodic IN NAK Effective Mask (GINNakEffMsk)	0x0
NPTxFEmpMsk	[5]	R/W	Mode: Host and Device Non-periodic TxFIFO Empty Mask (NPTxFEmpMsk)	0x0
RxFLvIMsk	[4]	R/W	Mode: Host and Device Receive FIFO Non-Empty Mask (RxFLvIMsk)	0x0

SofMsk	[3]	R/W	Mode: Host and Device Start of (micro)Frame Mask (SofMsk)	0x0
OTGIntMsk	[2]	R/W	Mode: Host and Device OTG Interrupt Mask (OTGIntMsk)	0x0
ModeMisMsk	[1]	R/W	Mode: Host and Device Mode Mismatch Interrupt Mask (ModeMisMsk)	0x0
RSVD	[0]	-	RESERVED	-

#### 7.4.9 GRXSTSR

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x1C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[30:25]	-	RESERVED	-
FN	[24:21]	R	Mode: Device only Frame Number (FN) This is the least significant 4 bits of the (micro)Frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.	0x0
PktSts	[20:17]	R	Mode: Host only Packet Status (PktSts) Indicates the status of the received packet 4'b0010: IN data packet received 4'b0011: IN transfer completed (triggers an interrupt) 4'b0101: Data toggle error (triggers an interrupt) 4'b0111: Channel halted (triggers an interrupt) Others: Reserved Mode: Device only Packet Status (PktSts) Indicates the status of the received packet 4'b0001: Global OUT NAK (triggers an interrupt) 4'b0010: OUT data packet received 4'b0011: OUT transfer completed (triggers an interrupt) 4'b0100: SETUP transaction completed (triggers an interrupt) 4'b0110: SETUP data packet received Others: Reserved	0x0
DPID	[16:15]	R	Mode: Host only Data PID (DPID) Indicates the Data PID of the received packet 2'b00: DATA0 2'b10: DATA1 2'b01: DATA2 2'b11: MDATA Mode: Device only Data PID (DPID) Indicates the Data PID of the received OUT data packet 2'b00: DATA0 2'b10: DATA1 2'b01: DATA2 2'b11: MDATA	0x0
BCnt	[14:4]	R	Mode: Host only Byte Count (BCnt) Indicates the byte count of the received IN data packet. Mode: Device only Byte Count (BCnt) Indicates the byte count of the received data packet.	0x0
ChNum	[3:0]	R	Mode: Host only Channel Number (ChNum) Indicates the channel number to which the current received packet belongs. Mode: Device only Endpoint Number (EPNum) Indicates the endpoint number to which the current received packet belongs.	0x0

#### 7.4.10 GRXSTSP

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x20 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[30:25]	-	RESERVED	-
FN	[24:21]	R	Mode: Device only Frame Number (FN) This is the least significant 4 bits of the (micro)Frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.	0x0
PktSts	[20:17]	R	Mode: Host only Packet Status (PktSts) Indicates the status of the received packet 4'b0010: IN data packet received 4'b0011: IN transfer completed (triggers an interrupt) 4'b0101: Data toggle error (triggers an interrupt) 4'b0111: Channel halted (triggers an interrupt) Others: Reserved Mode: Device only Packet Status (PktSts) Indicates the status of the received packet 4'b0001: Global OUT NAK (triggers an interrupt) 4'b0010: OUT data packet received 4'b0111: OUT transfer completed (triggers an interrupt) 4'b100: SETUP transaction completed (triggers an interrupt) 4'b0110: SETUP data packet received Others: Reserved	0x0
DPID	[16:15]	R	Mode: Host only Data PID (DPID) Indicates the Data PID of the received packet 2'b00: DATA0 2'b10: DATA1 2'b01: DATA2 2'b11: MDATA Mode: Device only Data PID (DPID) Indicates the Data PID of the received OUT data packet 2'b00: DATA0 2'b10: DATA1 2'b01: DATA2 2'b11: MDATA	0x0
BCnt	[14:4]	R	Mode: Host only Byte Count (BCnt) Indicates the byte count of the received IN data packet. Mode: Device only Byte Count (BCnt) Indicates the byte count of the received data packet.	0x0
ChNum	[3:0]	R	Mode: Host only Channel Number (ChNum) Indicates the channel number to which the current received packet belongs. Mode: Device only Endpoint Number (EPNum) Indicates the endpoint number to which the current received packet belongs.	0x0

### 7.4.11 GRXFSIZ

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x24 Reset Value = 0x16CE

Name	Bit	Type	Description	Reset Value
RSVD	[25:13]	-	RESERVED	-
RxFDep	[12:0]	R/W	<p>Mode: Host and Device RxFIFO Depth (RxFDep)  This value is in terms of 32-bit words.  Minimum value is 16  Maximum value is 32,768  The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth during configuration. If Enable Dynamic FIFO Sizing was selected, you can write a new value in this field. Programmed values must not exceed the power-on value</p>	0x16CE

### 7.4.12 GNPTXFSIZ

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x28 Reset Value = 0x40016CE

Name	Bit	Type	Description	Reset Value
NPTXFDep	[31:16]	R/W	<p>Mode: Host only  Non-periodic TxFIFO Depth (NPTxFDep)  For host mode, this field is always valid.  For Device mode, this field is valid for shared fifo  This value is in terms of 32-bit words.  Minimum value is 16  Maximum value is 32,768  Programmed values must not exceed the power-on value.  Mode: Device only  IN Endpoint TxFIFO 0 Depth (INEPTxF0Dep)  This value is in terms of 32-bit words.  Minimum value is 16  Maximum value is 32,768  This field is determined by Enable Dynamic FIFO Sizing  Programmed values must not exceed the power-on value.</p>	0x400
NPTXFStAddr	[15:0]	R/W	<p>Mode: Host only  Non-periodic Transmit RAM Start Address (NPTxFStAddr)  For host mode, this field is always valid. This field contains the memory start address  For Non-periodic Transmit FIFO RAM.  Programmed values must not exceed the power-on value.  Mode: Device only  IN Endpoint FIFO0 Transmit RAM Start Address (INEPTxF0StAddr)  This field contains the memory start address For IN Endpoint Transmit FIFO# 0.  Programmed values must not exceed the power-on value.</p>	0x16CE

#### 7.4.13 GNPTXSTS

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x2C Reset Value = 0x80400

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
NPTxQTop	[30:24]	R	<p>Top of the Non-periodic Transmit Request Queue (NPTxQTop)            Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC.</p> <p>Bits [30:27]: Channel/endpoint number</p> <p>Bits [26:25]:</p> <ul style="list-style-type: none"> <li>- 2'b00: IN/OUT token</li> <li>- 2'b01: Zero-length transmit packet (device IN/host OUT)</li> <li>- 2'b10: PING/CSPLIT token</li> <li>- 2'b11: Channel halt command</li> </ul> <p>Bit [24]: Terminate (last Entry For selected channel/endpoint)</p>	0x0
NPTxQSpAvail	[23:16]	R	<p>Non-periodic Transmit Request Queue Space Available (NPTxQSpAvail)</p> <p>Indicates the amount of free space available in the Non-periodic Transmit Request Queue. This queue holds both IN and OUT requests in Host mode. Device mode has only IN requests.</p> <p>8'h0: Non-periodic Transmit Request Queue is full</p> <p>8'h1: 1 location available</p> <p>8'h2: 2 locations available</p> <p>n: n locations available (0 &lt;= n &lt;= 8)</p> <p>Others: Reserved</p>	0x8
NPTxFSpAvail	[15:0]	R	<p>Non-periodic TxFIFO Space Avail (NPTxFSpAvail)</p> <p>Indicates the amount of free space available in the Non-periodic TxFIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Non-periodic TxFIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt;= n &lt;= 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x400

#### 7.4.14 GPIO

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x38 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
GPO	[31:16]	R/W	<p>General Purpose Output (GPO)</p> <p>This field is driven as an output from the core, gp_o[15:0]. The application can program this field to determine the corresponding value on the gp_o[15:0] output.</p>	0x0
GPI	[15:0]	R	<p>General Purpose Input (GPI)</p> <p>This field's read value reflects the gp_i[15:0] core input value.</p>	0x0

#### 7.4.15 GUID

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x3C Reset Value = 0x12345678

Name	Bit	Type	Description	Reset Value
GUID	[31:0]	R	User ID (UserID) Application-programmable ID field. Reset: Configurable	0x12345678

#### 7.4.16 GSNPSSID

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x40 Reset Value = 0x4F54330A

Name	Bit	Type	Description	Reset Value
SynopsysID	[31:0]	R	Release number of the DWC_otg core being used currently OTG	0x4F54330A

#### 7.4.17 GHWCFG1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x44 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EpDir	[31:0]	R	This 32-bit field uses two bits per endpoint to determine the endpoint direction. Endpoint Bits [31:30]: Endpoint 15 direction Bits [29:28]: Endpoint 14 direction ... Bits [3:2]: Endpoint 1 direction Bits[1:0]: Endpoint 0 direction (always BIDIR) Direction 2'b00: BIDIR (IN and OUT) endpoint 2'b01: IN endpoint 2'b10: OUT endpoint 2'b11: Reserved	0x0

### 7.4.18 GHWCFG2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x48 Reset Value = 0x208FFC50

Name	Bit	Type	Description	Reset Value
TknQDepth	[30:26]	R	Device Mode IN Token Sequence Learning Queue Depth (TknQDepth) Range: 0-30	0x8
PTxQDepth	[25:24]	R	Host Mode Periodic Request Queue Depth (PTxQDepth) 2'b00: 2 2'b01: 4 2'b10: 8 2'b11:16	0x0
NPTxQDepth	[23:22]	R	Non-periodic Request Queue Depth (NPTxQDepth) 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved	0x2
RSVD	[21]	-	RESERVED	-
MultiProcIntrpt	[20]	R	Multi Processor Interrupt Enabled (MultiProcIntrpt) 1'b0: No 1'b1: Yes	0x0
DynFifoSizing	[19]	R	Dynamic FIFO Sizing Enabled (DynFifoSizing) 1'b0: No 1'b1: Yes	0x1
PerioSupport	[18]	R	Periodic OUT Channels Supported in Host Mode (PerioSupport) 1'b0: No 1'b1: Yes	0x1
NumHstChnl	[17:14]	R	Number of Host Channels (NumHstChnl) Indicates the number of host channels supported by the core in Host mode. The range of this field is 0-15: 0 specifies 1 channel, 15 specifies 16 channels.	0xF
NumDevEps	[13:10]	R	Number of Device Endpoints (NumDevEps) Indicates the number of device endpoints supported by the core in Device mode in addition to control endpoint 0. The range of this field is 1-15.	0xF
FSPhyType	[9:8]	R	Full-Speed PHY Interface Type (FSPhyType) 2'b00: Full-speed interface not supported 2'b01: Dedicated full-speed interface 2'b10: FS pins shared with UTMI+ pins 2'b11: FS pins shared with ULPI pins	0x0
HSPhyType	[7:6]	R	High-Speed PHY Interface Type (HSPhyType) 2'b00: High-Speed interface not supported 2'b01: UTMI+ 2'b10: ULPI 2'b11: UTMI+ and ULPI	0x1
SingPnt	[5]	R	Point-to-Point (SingPnt) 1'b0: Multi-point application (hub and split support) 1'b1: Single-point application (no hub and split support)	0x0
OtgArch	[4:3]	R	Architecture (OtgArch) 2'b00: Slave-Only 2'b01: External DMA 2'b10: Internal DMA Others: Reserved	0x2
OtgMode	[2:0]	R	Mode of Operation (OtgMode) 3'b000: HNP- and SRP-Capable OTG (Host & Device) 3'b001: SRP-Capable OTG (Host & Device) 3'b010: Non-HNP and Non-SRP Capable OTG (Host & Device) 3'b011: SRP-Capable Device 3'b100: Non-OTG Device 3'b101: SRP-Capable Host 3'b110: Non-OTG Host Others: Reserved	0x0

### 7.4.19 GHWCFG3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x4C Reset Value = 0x164EF0E8

Name	Bit	Type	Description	Reset Value
DfifoDepth	[31:16]	R	DFIFO Depth (DfifoDepth - EP_LOC_CNT) This value is in terms of 32-bit words. Minimum value is 32 Maximum value is 32,768	0x164E
LPMMode	[15]	R	LPM mode specified for Mode of Operation.	0x1
BCSupport	[14]	R	This bit indicates the HS OTG controller support for Battery Charger. 0 - No Battery Charger Support 1 - Battery Charger support present.	0x1
HSICMode	[13]	R	HSIC mode specified for Mode of Operation Value Range: 0 - 1 1: HSIC-capable with shared UTMI PHY interface 0: Non-HSIC-capable	0x1
ADPSupport	[12]	R	This bit indicates whether ADP logic is present within or external to the HS OTG controller 0: No ADP logic present with DWC_otg controller 1: ADP logic is present along with DWC_otg controller.	0x1
RstType	[11]	R	Reset Style For Clocked always Blocks in RTL (RstType) 1'b0: Asynchronous reset is used in the core 1'b1: Synchronous reset is used in the core	0x0
OptFeature	[10]	R	Optional Features Removed (OptFeature) Indicates whether the User ID register, GPIO interface ports, and SOF toggle and counter ports were removed for gate count optimization by enabling Remove Optional Features. 1'b0: No 1'b1: Yes	0x0
VndctlSupt	[9]	R	Vendor Control Interface Support (VndctlSupt) 1'b0: Vendor Control Interface is not available on the core. 1'b1: Vendor Control Interface is available.	0x0
I2CIntSel	[8]	R	I2C Selection (I2CIntSel) 1'b0: I2C Interface is not available on the core. 1'b1: I2C Interface is available on the core.	0x0
OtgEn	[7]	R	OTG Function Enabled (OtgEn) The application uses this bit to indicate the DWC_otg core's OTG capabilities. 1'b0: Not OTG capable 1'b1: OTG Capable	0x1
PktSizeWidth	[6:4]	R	Width of Packet Size Counters (PktSizeWidth) 3'b000: 4 bits 3'b001: 5 bits 3'b010: 6 bits 3'b011: 7 bits 3'b100: 8 bits 3'b101: 9 bits 3'b110: 10 bits Others: Reserved	0x6
XferSizeWidth	[3:0]	R	Width of Transfer Size Counters (XferSizeWidth) 4'b0000: 11 bits 4'b0001: 12 bits ... 4'b1000: 19 bits Others: Reserved	0x8

### 7.4.20 GHWCFG4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x50 Reset Value = 0xFFFF08030

Name	Bit	Type	Description	Reset Value
DescDMA	[31]	R	Scatter/Gather DMA configuration 1'b0: Non Dynamic configuration 1'b1: Dynamic configuration	0x1
DescDMAEnabled	[30]	R	Scatter/Gather DMA configuration 1'b0: Non-Scatter/Gather DMA configuration 1'b1: Scatter/Gather DMA configuration	0x1
INEps	[29:26]	R	Number of Device Mode IN Endpoints Including Control Endpoints (INEps) Range 0 -15 0 : 1 IN Endpoint 1 : 2 IN Endpoints ... 15 : 16 IN Endpoints	0xF
DedFifoMode	[25]	R	Enable Dedicated Transmit FIFO For device IN Endpoints (DedFifoMode) 1'b0 : Dedicated Transmit FIFO Operation not enabled. 1'b1 : Dedicated Transmit FIFO Operation enabled.	0x1
SessEndFltr	[24]	R	session_end Filter Enabled (SessEndFltr) 1'b0: No filter 1'b1: Filter	0x1
BValidFltr	[23]	R	b_valid Filter Enabled (BValidFltr) 1'b0: No filter 1'b1: Filter	0x1
AValidFltr	[22]	R	a_valid Filter Enabled (AValidFltr) 1'b0: No filter 1'b1: Filter	0x1
VBusValidFltr	[21]	R	VBUS Valid Filter Enabled (VBusValidFltr) 1'b0: No filter 1'b1: Filter	0x1
IddgFltr	[20]	R	IDDG Filter Enable (IddgFltr) 1'b0: No filter 1'b1: Filter	0x1
NumCtlEps	[19:16]	R	Number of Device Mode Control Endpoints in Addition to Endpoint 0 (NumCtlEps) Range: 0-15	0x0
PhyDataWidth	[15:14]	R	UTMI+ PHY/ULPI-to-Internal UTMI+ Wrapper Data Width (PhyDataWidth) When a ULPI PHY is used, an internal wrapper converts ULPI to UTMI+. 2'b00: 8 bits 2'b01: 16 bits 2'b10: 8/16 bits, software selectable Others: Reserved	0x2
RSVD	[13:8]	-	RESERVED	-
ExtendedHibernation	[7]	R	Enable Hibernation 1'b0: Extended Hibernation feature not enabled 1'b1: Extended Hibernation feature enabled	0x0
Hibernation	[6]	R	Enable Hibernation (Hibernation) 1'b0: Hibernation feature not enabled 1'b1: Hibernation feature enabled	0x0
AhbFreq	[5]	R	Minimum AHB Frequency Less Than 60 MHz (AhbFreq) 1'b0: No 1'b1: Yes	0x1
PartialPwrDn	[4]	R	Enable Partial Power Down (PartialPwrDn) 1'b0: Partial Power Down Not Enabled 1'b1: Partial Power Down Enabled	0x1
NumDevPerioEps	[3:0]	R	Number of Device Mode Periodic IN Endpoints (NumDevPerioEps) Range: 0-15	0x0

### 7.4.21 GLPMCFG

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x54 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
InvSelHsic	[31]	R/W	<p>HSIC-Invert Select HSIC (InvSelHsic)</p> <p>The application uses this bit to control the DWC_otg core HSIC enable/disable. This Bit overrides and functionally inverts the if_sel_hsic input port signal. If the core operates as non-HSIC-capable, it can then connect to only PHYs which are not HSIC capable. If the core operates as HSIC-capable, it can then connect to only PHYs which are HSIC capable.</p> <p>If if_sel_hsic input signal is 1, InvSelHsic = 1b1: HSIC capability is not enabled, InvSelHsic = 1b0: HSIC capability is enabled.</p> <p>If if_sel_hsic input signal is 0, InvSelHsic = 1b1: HSIC capability is enabled, InvSelHsic = 1b0: HSIC capability is not enabled.</p>	0x0
HSICCon	[30]	R/W	<p>HSIC-Connect (HSICCon)</p> <p>The application must use this bit to initiate HSIC ATTACH sequence.</p> <p>Host Mode: Once this bit is set, the Host Core configures to drive HSIC IDLE state (STROBE=1&amp;DATA=0) on the BUS. It then waits for device to initiate CONNECT.</p> <p>Device Mode: Once this bit is set, the Device Core waits for HSIC IDLE Linestate on the BUS.</p> <p>Upon receiving the IDLE linestate it then initiates HSIC CONNECT.</p>	0x0
LPM_RestoreSlpsts	[29]	R/W	<p>LPM Restore Sleep Status (LPM_RestoreSlpsts)</p> <p>When the application power gates the core (Partial Power Down / Hibernation) the application needs to program this bit to restore the LPM status in the core.</p> <p>The application needs to program this bit, during restore process, based on whether it had decided to go into Shallow Sleep (Clock Gating Only) or Deep Sleep (Power Gating) based on the BESL value received from the Host</p> <p>1'b0: The application puts the core in Shallow Sleep based on BESL value from the Host</p> <p>1'b1: The application puts the core in Deep Sleep based on BESL value from the Host</p>	0x0
LPM_EnBESL	[28]	R/W	<p>LPM Enable BESL (LPM_EnBESL)</p> <p>This bit enables the BESL feature as defined in LPM Errata</p> <p>1'b0: The core works as per USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification as of July 16, 2007</p> <p>1'b1: The core works as per the LPM Errata</p>	0x0
LPM_RetryCnt_Sts	[27:25]	R	<p>LPM Retry Count Status (LPM_RetryCnt_Sts)</p> <p>Number of LPM Host Retries still remaining to be transmitted for the current LPM sequence.</p>	0x0
SndLPM	[24]	R/W	<p>Send LPM Transaction (SndLPM)</p> <p>Host Mode:</p> <p>When set by application software, an LPM transaction containing two tokens, EXT &amp; LPM is sent. Cleared by the hardware once a valid response (STALL./NYET/ACK) is received from the Device or the core had finished transmitting the programmed number of LPM retries. Note that this bit should be set only when the host is connected to local port.</p>	0x0
LPM_Retry_Cnt	[23:21]	R/W	<p>LPM Retry Count (LPM_Retry_Cnt)</p> <p>Number of additional LPM retries that the HOST would perform if the Device Response was an ERROR until a valid device response is received (STALL./NYET/ACK).</p>	0x0

LPM_Chnl_Idx	[20:17]	R/W	<p>LPM Channel Index            The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and end point number programmed in the corresponding channel into the LPM transaction.</p>	0x0
L1ResumeOK	[16]	R	<p>Sleep State Resume OK (L1ResumeOK)            Indicates that the application or host can start resume from Sleep state.            This bit is valid in LPM sleep (L1) state.            It is set in sleep mode after a delay of 50 micro sec (TL1Residency).            The bit is reset when SlpSts is 0            1b0: The application/core cannot start resume from Sleep state.            1b1: The application/core can start resume from Sleep state.</p>	0x0
SlpSts	[15]	R	<p>Port Sleep Status (SlpSts)            Device Mode: This bit is set as long as a Sleep condition is present on the USB bus.            The core enters the Sleep state when an ACK response is sent to an LPM transaction and the expiry of timer TL1TokenRetry.            The core comes out of sleep:            - When there is any activity on the USB line_state            - When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig) or when the application resets or soft-disconnects the device.            Host Mode: The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device.            The read value of this bit reflects the current Sleep status of the port.            The core clears this bit after:            - The core detects a remote L1 Wakeup signal;            - The application sets the Port Reset bit or the Port L1Resume bit in the HPRT register; or            - The application sets the L1Resume/ Remote Wakeup Detected Interrupt bit or Disconnect Detected Interrupt bit in the Core Interrupt register (GINTSTS.L1WkUpInt or GINTSTS.DisconnInt, respectively).            1b0: Core not in L1.            1b1: Core in L1.</p>	0x0
CoreL1Res	[14:13]	R	<p>LPM response (CoreL1Res)            Device Mode: The response of the core to LPM transaction received is reflected in these two bits.            Host Mode: Handshake response received from local device for LPM transaction            11 - ACK            10 - NYET            01 - STALL            00 - ERROR (No handshake response)</p>	0x0

HIRD_Thres	[12:8]	R/W	<p>BESL/HIRD Threshold (HIRD_Thres)  Device Mode:  EnBESL = 1'b0  The core puts the PHY into deep low power mode in L1  (by core asserting L1SuspendM) when HIRD value is greater than  or  equal to the value defined in this field HIRD_Thres[3:0] and HIRD_Thres[4] is set to 1'b1.  EnBESL = 1'b1  The core puts the PHY into deep low power mode in L1 (by core asserting L1SuspendM)  when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0]  and BESL_Thres [4] is set to 1'b1.</p> <p>Host Mode:  The core puts the PHY into deep low power mode in L1 (by core asserting L1SuspendM)  when HIRD_Thres[4] is set to 1'b1. HIRD_Thres[3:0] specifies the time  for which resume signaling is to be reflected by host  (TL1HubDrvResume2) on the USB bus  when it detects device initiated resume. When EnBESL = 1'b0  HIRD_Thres should not be  programmed with a value greater than 4'b1100 in host mode since  this will exceed maximum  TL1HubDrvResume2. When EnBESL = 1'b1 HIRD_Thres should  not be programmed with a value  greater than 4'b0110 in host mode since this will exceed maximum  TL1HubDrvResume2.</p>	0x0
EnbISlpM	[7]	R/W	<p>Enable utmi_sleep_n (EnbISlpM)  For ULPI interface: The application uses this bit to write to the function control [7] in the L1 state, to enable the PHY to go into Low Power mode. For the host, this bit is valid only in Local Device mode.</p> <ul style="list-style-type: none"> <li>- 1'b0: Writes to the ULPI Function Control Bit[7] are disabled.</li> <li>- 1'b1: The core is enabled to write to the ULPI Function Control Bit[7], which enables the PHY to enter Low-Power mode.</li> </ul> <p>Note: When a ULPI interface is configured, enabling this bit results in a write to Bit 7 of the ULPI Function Control register. The Synopsis ULPI PHY supports writing to this bit, and in the L1 state asserts SleepM when utmi_l1_suspend_n cannot be asserted.</p> <p>For all other interfaces: The application uses this bit to control utmi_sleep_n assertion to the PHY in the L1 state. For the host, this bit is valid only in Local Device mode.</p> <ul style="list-style-type: none"> <li>- 1'b0: utmi_sleep_n assertion from the core is not transferred to the external PHY.</li> <li>- 1'b1: utmi_sleep_n assertion from the core is transferred to the external PHY when utmi_l1_suspend_n cannot be asserted</li> </ul>	0x0
bRemoteWake	[6]	R/W	<p>RemoteWakeEnable (bRemoteWake)  Host Mode: The value of remote wake up to be sent in wIndex field of LPM transaction.</p> <p>Device Mode (Read-Only): This field is updated with the Received LPM Token bRemoteWake bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction.</p>	0x0

HIRD	[5:2]	R/W	<p>Host Initiated Resume Duration (HIRD) EnBESL = 1'b0 Host Initiated Resume Duration Host Mode: The value of HIRD to be sent in an LPM transaction. This value is also used to initiate resume for a duration TL1HubDrvResume1 for host initiated resume Device Mode(Read-Only): This field is updated with the Received LPM Token HIRD bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction.</p> <p>EnBESL = 1'b1 Best Effort Service Latency (BESL) Host Mode: The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration TL1HubDrvResume1 for host initiated resume Device Mode: This field is updated with the Received LPM Token BESL bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction</p>	0x0
AppL1Res	[1]	R/W	<p>Mode: Device only LPM response programmed by application (AppL1Res) Handshake response to LPM token pre-programmed by device application software. The response depends on GLPM-CFG.LPMCap. If GLPMCFG.LPMCap is 1'b0, the core always responds with a NYET. If GLPMCFG.LPMCap is 1'b1, the core responds as follows:  <ul style="list-style-type: none"> <li>- 1: ACK</li> <li>Even though an ACK is pre-programmed, the core responds with an ACK only on a successful LPM transaction. The LPM transaction is successful if:</li> <li>- There are no PID/CRC5 errors in both the EXT token and the LPM token (else ERROR)</li> <li>- A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL)</li> <li>- No data is pending in the Transmit queue (else NYET)</li> <li>- 0: NYET</li> </ul> The pre-programmed software bit is overridden for response to LPM token when: <ul style="list-style-type: none"> <li>- The received bLinkState is not L1 (STALL response)</li> <li>- An error is detected in either of the LPM token packets due to corruption (ERROR response).</li> </ul> </p>	0x0
LPMCap	[0]	R/W	<p>LPM-Capable (LPMCap) The application uses this bit to control the DWC_otg core LPM capabilities. If the core operates as a non-LPM-capable host, it cannot request the connected device/hub to activate LPM mode. If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions. 1b0: LPM capability is not enabled. 1b1: LPM capability is enabled.</p>	0x0

### 7.4.22 GPWRDN

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x58 Reset Value = 0x10

Name	Bit	Type	Description	Reset Value
MultValldBC	[28:24]	R	MultValldBC Battery Charger ACA inputs in the following order: Bit 26 - rid_float. Bit 25 - rid_gnd Bit 24 - rid_a Bit 23 - rid_b Bit 22 - rid_c These bits are present only if BC_SUPPORT = 1. Otherwise, these bits are reserved and will read 5'h0.	0x0
ADPInt	[23]	R/W	ADP Interrupt (ADPInt) This bit is set whenever there is a ADP event	0x0
BSessVld	[22]	R	B Session Valid (BSessVld) This field reflects the B session valid status signal from the PHY. 1'b0: B-Valid is 0. 1'b1: B-Valid is 1. This bit is valid only when GPWRDN.PMUActv is 1.	0x0
IDDIG	[21]	R	This bit indicates the status of the signal IDDIG. The application must read this bit after receiving GPWRDN.StsChngInt and decode based on the previous value stored by the application. Indicates the current mode. 1'b0: Host mode 1'b1: Device mode This bit is valid only when GPWRDN.PMUActv is 1.	0x0
LineState	[20:19]	R	LineState This field indicates the current linestate on USB as seen by the PMU module. 2'b00: DM = 0, DP = 0. 2'b01: DM = 0, DP = 1. 2'b10: DM = 1, DP = 0. 2'b11: Not-defined. This bit is valid only when GPWRDN.PMUActv is 1.	0x0
StsChngIntMsk	[18]	R/W	StsChngIntMsk Mask For StsChng Interrupt	0x0
StsChngInt	[17]	R/W	Status Change Interrupt (StsChngInt) This field indicates a status change in either the IDDIG or BSessVld signal. 1'b0: No Status change 1'b1: Status change detected After receiving this interrupt the application should read the GPWRDN register and interpret the change in IDDIG or BSesVld with respect to the previous value stored by the application. Note: When Battery Charger is enabled and the ULPI interface is used, if StsChngInt is received and the application reads GPWRDN register and determines that it is because of a change in the value of IDDIG, then StsChngInt may be generated once again within the next few clock cycles. This occurs because of an ambiguity in the implementation of Battery Charger Support over the ULPI interface. After receiving the StsChngInt for the second time the application can once again read the GPWRDN register. However, this time the value IDDIG (or BSesVld) will not have changed. The application then processes the second interrupt but no further action will be required as a result.	0x0
SRPDetectMsk	[16]	R/W	SRPDetectMsk Mask For SRPDetect Interrupt	0x0
SRPDetect	[15]	R/W	SRPDetect This field indicates that SRP has been detected by the PMU. This field generates an interrupt. After detecting SRP during hibernation the application should not restore the core. The application should get into the initialization process. 1'b0: SRP not detected 1'b1: SRP detected	0x0

DisableVBUS	[6]	R/W	<p><b>DisableVBUS</b>  The application should program this bit if HPRT0.PrtPwr was programmed to 0 before entering Hibernation. This is to indicate PMU whether session was ended before entering Hibernation.  1'b0: HPRT0.PrtPwr was not programmed to 0.  1'b1: HPRT0.PrtPwr was programmed to 0.</p> <p><b>Device Mode:</b>  The application must program this bit to inform the PMU whether the bvalid valid signal is high (session valid) or low (session end) whenever the core is switched off.  1'b0: bvalid signal is High (Session Valid)  1'b1: bvalid signal is Low (Session End)  Note: This bit is valid only when GPWRDN.PMUActv is 1.</p>	0x0
PwrDnSwtch	[5]	R/W	<p><b>Power Down Switch (PwrDnSwtch)</b>  This bit indicates to the DWC_otg core VDD switch is in ON/OFF state  1'b0: DWC_otg is in ON state  1'b1: DWC_otg is in OFF state  Note: This bit must not be written to during normal mode of operation.</p>	0x0
PwrDnRst_n	[4]	R/W	<p><b>Power Down ResetN (PwrDnRst_n)</b>  The application must program this bit to reset the DWC OTG core during the Hibernation exit process or during ADP when powering up the core (in case the DWC OTG core was powered off during ADP process).  1'b1: DWC_otg is in normal operation  1'b0: reset DWC_otg  Note: This bit must not be written to during normal mode of operation.</p>	0x1
PwrDnClmp	[3]	R/W	<p><b>Power Down Clamp (PwrDnClmp)</b>  The application must program this bit to enable or disable the clamps to all the outputs of the DWC OTG core module to prevent the corruption of other active logic.  1'b0: Disable PMU power clamp  1'b1: Enable PMU power clamp</p>	0x0
PMUActv	[1]	R/W	<p><b>PMU Active (PMUActv)</b>  This bit is to enable or disable the PMU logic.  1'b0: Disable PMU module  1'b1: Enable PMU module  Note: This bit must not be written to during normal mode of operation.</p>	0x0
PMUIntSel	[0]	R/W	<p><b>PMU Interrupt Select (PMUIntSel)</b>  A write to this bit with 1'b1 enables the PMU to generate interrupts to the application. During this state all interrupts from the DWC_otg_core module are blocked to the application.  Note: This bit must be set to 1'b1 before the core is put into hibernation  1'b0: Internal DWC_otg_core interrupt is selected  1'b1: External DWC_otg_pmu interrupt is selected  Note: This bit must not be written to during normal mode of operation.</p>	0x0

### 7.4.23 GDFIFO CFG

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x5C Reset Value = 0x164E16CE

Name	Bit	Type	Description	Reset Value
EPIInfoBaseAddr	[31:16]	R/W	EPIInfoBaseAddr This field provides the start address of the EP info controller.	0x164E
GDFIFO Cfg	[15:0]	R/W	GDFIFO Cfg This field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non zero value to this register. The value programmed must conform to the guidelines described in 'FIFO RAM Allocation'. The DWC_otg core does not have any corrective logic if the FIFO sizes are programmed incorrectly.	0x16CE

### 7.4.24 GADPCTL

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x60 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
AR	[28:27]	R/W	Access Request (AR) 2'b00 Read/Write Valid (updated by the core) 2'b01 Read 2'b10 Write 2'b11 - Reserved	0x0
AdpToutMsk	[26]	R/W	ADP Timeout Interrupt Mask (AdpToutMsk) When this bit is set, it unmasks the interrupt because of AdpTmouln. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).	0x0
AdpSnsIntMsk	[25]	R/W	ADP Sense Interrupt Mask (AdpSnsIntMsk) When this bit is set, it unmasks the interrupt due to AdpSnsInt. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).	0x0
AdpPrbIntMsk	[24]	R/W	ADP Probe Interrupt (AdpPrbInt) This bit is relevant only for an ADP probe. When this bit is set, it means that the ramp time has completed (GADPCTL.RTIM has reached its terminal value of 0x7FF). This is a debug feature that allows software to read the ramp time after each cycle. This bit is valid only if OTGVer = 1'b1.	0x0
AdpToutInt	[23]	R/W	ADP Timeout Interrupt (AdpToutInt) This bit is relevant only for an ADP probe. When this bit is set, it means that the ramp time has completed (GADPCTL.RTIM has reached its terminal value of 0x7FF). This is a debug feature that allows software to read the ramp time after each cycle. This bit is valid only if OTGVer = 1'b1.	0x0
AdpSnsInt	[22]	R/W	ADP Sense Interrupt (AdpSnsInt) When this bit is set, it means that the VBUS voltage is greater than VadpSns value or VadpSns is reached. This bit is valid only if OTGVer = 1'b1 (GOTGCTL[20]).	0x0
AdpPrbInt	[21]	R/W	ADP Probe Interrupt (AdpPrbInt) When this bit is set, it means that the VBUS voltage is greater than VADP_PRB or VadpPrb is reached. This bit is valid only if OTGVer = 1'b1 (GOTGCTL[20]).	0x0

ADPEn	[20]	R/W	<p>ADP Enable (ADPEn)</p> <p>When set, the core performs either ADP probing or sensing based on EnaPrb or EnaSns.</p> <p>This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).</p>	0x0
ADPRes	[19]	R/W	<p>ADP Reset (ADPRes)</p> <p>When set, ADP controller is reset. This bit is auto-cleared after the reset procedure is complete in ADP controller.</p> <p>This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).</p>	0x0
EnaSns	[18]	R/W	<p>Enable Sense (EnaSns)</p> <p>When programmed to 1'b1, the core performs a sense operation.</p> <p>This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).</p>	0x0
EnaPrb	[17]	R/W	<p>Enable Probe (EnaPrb)</p> <p>When programmed to 1'b1, the core performs a probe operation.</p> <p>This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).</p>	0x0
RTIM	[16:6]	R	<p>RAMP TIME (RTIM)</p> <p>These bits capture the latest time it took for VBUS to ramp from VADP_SINK to VADP_PRB. The bits are defined in units of 32 kHz clock cycles as follows:</p> <p>0x000 - 1 cycles      0x001 - 2 cycles      0x002 - 3 cycles      and so on till      0x7FF - 2048 cycles</p> <p>A time of 1024 cycles at 32 kHz corresponds to a time of 32 msec.      (Note for scaledown ramp_timeout =      prb_delta == 2'b00 =&gt; 200 cycles.      prb_delta == 2'b01 =&gt; 100 cycles.      prb_delta == 2'b01 =&gt; 50 cycles.      prb_delta == 2'b01 =&gt; 25 cycles.)</p>	0x0
PrbPer	[5:4]	R/W	<p>Probe Period (PrbPer)</p> <p>These bits sets the TadpPrd as follows:</p> <p>2'b00 - 0.625 to 0.925 sec (typical 0.775 sec)      2'b01 - 1.25 to 1.85 sec (typical 1.55 sec)      2'b10 - 1.9 to 2.6 sec (typical 2.275 sec)      2'b11 - Reserved      (PrbPer is also scaledown      prb_per== 2'b00 =&gt; 400 ADP clocks      prb_per== 2'b01 =&gt; 600 ADP clocks      prb_per== 2'b10 =&gt; 800 ADP clocks      prb_per==2'b11 =&gt; 1000 ADP clocks)</p>	0x0
PrbDelta	[3:2]	R/W	<p>Probe Delta (PrbDelta)</p> <p>These bits set the resolution for RTIM value. The bits are defined in units of 32 kHz clock cycles as follows:</p> <p>2'b00 - 1 cycles      2'b01 - 2 cycles      2'b10 - 3 cycles      2'b11 4 cycles</p> <p>For example if this value is chosen to 2'b01, it means that RTIM increments for every three 32Khz clock cycles.</p>	0x0
PrbDschg	[1:0]	R/W	<p>Probe Discharge (PrbDschg)</p> <p>These bits set the times for TADP_DSCHG. These bits are defined as follows:</p> <p>2'b00 4 msec (Scaledown 2 32Khz clock cycles)      2'b01 8 msec (Scaledown 4 32Khz clock cycles)      2'b10 16 msec (Scaledown 8 32Khz clock cycles)      2'b11 32 msec (Scaledown 16 32Khz clock cycles)</p>	0x0

#### 7.4.25 HPTXFSIZ

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x100 Reset Value = 0x16CE184E

Name	Bit	Type	Description	Reset Value
PTxFSize	[28:16]	R/W	Host Periodic TxFIFO Depth (PTxFSize) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest Host Mode Periodic Tx Data FIFO Depth. Programmed values must not exceed the power-on value.	0x16CE
PTxFStAddr	[12:0]	R/W	Host Periodic TxFIFO Start Address (PTxFStAddr) The power-on reset value of this register is the sum of the Largest Rx Data FIFO Depth and Largest Non-periodic Tx Data FIFO Depth Programmed values must not exceed the power-on value	0x184E

#### 7.4.26 DIEPTXF1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x104 Reset Value = 0x4001ACE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxFStAddr	[12:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxFStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0 < n < = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x1ACE

#### 7.4.27 DIEPTXF2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x108 Reset Value = 0x4001ECE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400

INEPnTxStAddr	[12:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x1ECE
---------------	--------	-----	---	--------

#### 7.4.28 DIEPTXF3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x10C Reset Value = 0x40022CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x22CE

#### 7.4.29 DIEPTXF4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x110 Reset Value = 0x40026CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x26CE

### 7.4.30 DIEPTXF5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x114 Reset Value = 0x4002ACE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFOOn (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x2ACE

### 7.4.31 DIEPTXF6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x118 Reset Value = 0x4002ECE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFOOn (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x2ECE

### 7.4.32 DIEPTXF7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x11C Reset Value = 0x40032CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400

INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x32CE
---------------	--------	-----	---	--------

#### 7.4.33 DIEPTXF8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x120 Reset Value = 0x40036CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x36CE

#### 7.4.34 DIEPTXF9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x124 Reset Value = 0x4003ACE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x3ACE

### 7.4.35 DIEPTXF10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x128 Reset Value = 0x4003ECE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[13:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFOOn (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x3ECE

### 7.4.36 DIEPTXF11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x12C Reset Value = 0x40042CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[14:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFOOn (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x42CE

### 7.4.37 DIEPTXF12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x130 Reset Value = 0x40046CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400

INEPnTxStAddr	[14:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x46CE
---------------	--------	-----	---	--------

#### 7.4.38 DIEPTXF13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x134 Reset Value = 0x4004ACE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[14:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x4ACE

#### 7.4.39 DIEPTXF14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x138 Reset Value = 0x4004ECE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxStAddr	[14:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x4ECE

#### 7.4.40 DIEPTXF15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x13C Reset Value = 0x40052CE

Name	Bit	Type	Description	Reset Value
INEPnTxFDep	[26:16]	R/W	IN Endpoint TxFIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth Programmed values must not exceed the power-on value	0x400
INEPnTxFStAddr	[14:0]	R/W	IN Endpoint FIFO Transmit RAM Start Address (INEPnTxFStAddr) This field contains the memory start address For IN endpoint Transmit FIFO (0<n< = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth Programmed values must not exceed the power-on value.	0x52CE

#### 7.4.41 DCFG

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x800 Reset Value = 0x8200000

Name	Bit	Type	Description	Reset Value
ResValid	[31:26]	R/W	Resume Validation Period (ResValid) This field is effective only when DCFG.Ena32KHzSusp is set. It will control the resume period when the core resumes from suspend. The core counts for ResValid number of clock cycles to detect a valid resume when this is set	0x2
PerSchIntvl	[25:24]	R/W	Periodic Scheduling Interval (PerSchIntvl) PerSchIntvl must be programmed only For Scatter/Gather DMA mode. Description: This field specifies the amount of time the Internal DMA engine must allocate For fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25,50 or 75% of (micro)frame. When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data . When no periodic endpoints are active, Then the internal DMA engine services non-periodic endpoints, ignoring this field. After the specified time within a (micro)frame, the DMA switches to fetching For non-periodic endpoints. 2'b00: 25% of (micro)frame. 2'b01: 50% of (micro)frame. 2'b10: 75% of (micro)frame. 2'b11: Reserved. Reset: 2'b00	0x0
DescDMA	[23]	R/W	Enable Scatter/gather DMA in device mode (DescDMA). When the Scatter/Gather DMA option selected during configuration of the RTL, the application can Set this bit during initialization to enable the Scatter/Gather DMA operation. NOTE: This bit must be modified only once after a reset. The following combinations are available For programming: GAHBCFG.DMAEn=0,DCFG.DescDMA=0 => Slave mode GAHBCFG.DMAEn=0,DCFG.DescDMA=1 => Invalid GAHBCFG.DMAEn=1,DCFG.DescDMA=0 => Buffered DMA mode GAHBCFG.DMAEn=1,DCFG.DescDMA=1 => Scatter/Gather DMA mode	0x0

RSVD	[17:16]	-	RESERVED	-
ErraticIntMsk	[15]	R/W	Erratic Error Interrupt Mask 1'b1: Mask early suspend interrupt on erratic error 1'b0: Early suspend interrupt is generated on erratic error	0x0
XCVRDLY	[14]	R/W	1'b1: Enable delay between xcvr_sel and txvalid during Device chirp 1'b0: No delay between xcvr_sel and txvalid during Device chirp	0x0
EnDevOutNak	[13]	R/W	Enable Device OUT NAK (EnDevOutNak) This bit enables setting NAK for Bulk OUT endpoints after the transfer is completed for Device mode Descriptor DMA 1'b0 : The core does not set NAK after Bulk OUT transfer complete 1'b1 : The core sets NAK after Bulk OUT transfer complete It is one time programmable after reset like any other DCFG register bits.	0x0
PerFrInt	[12:11]	R/W	Periodic Frame Interval (PerFrInt) Indicates the time within a (micro)frame at which the application must be notified using the End Of Periodic Frame Interrupt. This can be used to determine if all the isochronous traffic for that (micro)frame is complete. 2'b00: 80% of the (micro)frame interval 2'b01: 85% 2'b10: 90% 2'b11: 95%	0x0
DevAddr	[10:4]	R/W	Device Address (DevAddr) The application must program this field after every SetAddress control command.	0x0
Ena32KHzSusp	[3]	R/W	Enable 32 KHz Suspend mode (Ena32KHzSusp) This bit can be set only if FS PHY interface is selected. Else, this bit needs to be set to zero. When FS PHY interface is chosen and this bit is set, the core expects that the PHY clock during Suspend is switched from 48 MHz to 32 KHz.	0x0
NZStsOUTHShk	[2]	R/W	Non-Zero-Length Status OUT Handshake (NZStsOUTHShk) The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage. 1'b1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application. 1'b0: Send the received OUT packet to the application (zerolength or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.	0x0
DevSpd	[1:0]	R/W	Device Speed (DevSpd) Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected. 2'b00: High speed (USB 2.0 PHY clock is 30 MHz or 60 MHz) 2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz) 2'b10: Low speed (USB 1.1 transceiver clock is 6 MHz). If you select 6 MHz LS mode, you must do a soft reset. 2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)	0x0

### 7.4.42 DCTL

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x804 Reset Value = 0x2

Name	Bit	Type	Description	Reset Value
RSVD	[31:19]	-	RESERVED	-
DeepSleepBESLReject	[18]	R/W	1: Deep Sleep BESL Reject feature is enabled 0: Deep Sleep BESL Reject feature is disabled When enabled Core rejects LPM request with HIRD value greater than HIRD threshold programmed. NYET response is sent for LPM tokens with HIRD value greater than HIRD threshold. By default, the Deep Sleep BESL Reject feature is disabled. Default Value: 0h	0x0
EnContOnBNA	[17]	R/W	Enable Continue on BNA (EnContOnBNA) This bit enables the DWC_otg core to continue on BNA for Bulk OUT endpoints. With this feature enabled, when a Bulk OUT endpoint receives a BNA interrupt the core starts processing the descriptor that caused the BNA interrupt after the endpoint re-enables the endpoint. 1'b0: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the DOEPDMA descriptor. 1'b1: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the descriptor that received the BNA interrupt. This bit is valid only when OTG_EN_DESC_DMA == 1'b1. It is a one-time programmable after reset bit like any other DCTL register bits.	0x0
NakOnBble	[16]	R/W	NAK on Babble Error (NakOnBble) Set NAK automatically on babble (NakOnBble). The core sets NAK automatically for the endpoint on which babble is received.	0x0
IgnrFrmNum	[15]	R/W	Ignore Frame number For Isochronous End points (IgnrFrmNum) Do NOT program IgnrFrmNum bit to 1'b1 when the core is operating in threshold mode. Note: When Scatter/Gather DMA mode is enabled this feature is not applicable to High Speed, High bandwidth transfers. When this bit is enabled, there must be only one packet per descriptor. 0: The core transmits the packets only in the frame number in which they are intended to be transmitted. 1: The core ignores the frame number, sending packets immediately as the packets are ready. In Scatter/Gather DMA mode, if this bit is enabled, the packets are not flushed when a ISOIN token is received for an elapsed frame. When Scatter/Gather DMA mode is disabled, this field is used by the application to enable periodic transfer interrupt. The application can program periodic endpoint transfers for multiple (micro)frames. 0: periodic transfer interrupt feature is disabled, application needs to program transfers for periodic endpoints every (micro)frame 1: periodic transfer interrupt feature is enabled, application can program transfers for multiple (micro)frames for periodic endpoints. In non Scatter/Gather DMA mode the application will receive transfer complete interrupt after transfers for multiple (micro)frames are completed.	0x0

GMC	[14:13]	R/W	<p>Global Multi Count (GMC)            GMC must be programmed only once after initialization.            Applicable only For Scatter/Gather DMA mode. This indicates the number of packets to be serviced For that end point before moving to the next end point. It is only For non-periodic end points.            2'b00: Invalid.            2'b01: 1 packet.            2'b10: 2 packets.            2'b11: 3 packets.            When Scatter/Gather DMA mode is disabled, this field is reserved. and reads 2'b00.</p>	0x0
PWROnPrgDone	[11]	R/W	<p>Power-On Programming Done (PWROnPrgDone)            The application uses this bit to indicate that register programming is completed after a wake-up from Power Down mode.</p>	0x0
CGOUTNak	[10]	W	<p>Clear Global OUT NAK (CGOUTNak)            A write to this field clears the Global OUT NAK.</p>	0x0
SGOUTNak	[9]	W	<p>Set Global OUT NAK (SGOUTNak)            A write to this field sets the Global OUT NAK.            The application uses this bit to send a NAK handshake on all OUT endpoints.            The application must set the this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register (GINTSTS.GOUTNakEff) is cleared.</p>	0x0
CGNPIInNak	[8]	W	<p>Clear Global Non-periodic IN NAK (CGNPIInNak)            A write to this field clears the Global Non-periodic IN NAK.</p>	0x0
SGNPIInNak	[7]	W	<p>Set Global Non-periodic IN NAK (SGNPIInNak)            A write to this field sets the Global Non-periodic IN NAK. The application uses this bit to send a NAK handshake on all nonperiodic IN endpoints. The core can also Set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation.            The application must Set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (GINTSTS.GINNakEff) is cleared</p>	0x0
TstCtl	[6:4]	R/W	<p>Test Control (TstCtl)            3'b000: Test mode disabled            3'b001: Test_J mode            3'b010: Test_K mode            3'b011: Test_SE0_NAK mode            3'b100: Test_Packet mode            3'b101: Test_Force_Enable            Others: Reserved</p>	0x0
GOUTNaksts	[3]	R	<p>Global OUT NAK Status (GOUTNaksts)            1'b0: A handshake is sent based on the FIFO Status and the NAK and STALL bit settings.            1'b1: No data is written to the RxFIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.</p>	0x0
GNPINNaksts	[2]	R	<p>Global Non-periodic IN NAK Status (GNPINNaksts)            1'b0: A handshake is sent out based on the data availability in the transmit FIFO.            1'b1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.</p>	0x0

SftDiscon	[1]	R/W	<p>Soft Disconnect (SftDiscon)  The application uses this bit to signal the DWC_otg core to do a soft disconnect. As long as this bit is Set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.  The minimum duration For which the core must keep this bit Set is specified in Table 5-46.</p> <p>1'b0: Normal operation. When this bit is cleared after a soft disconnect, the core drives the phy_opmode_o signal on the UTMI+ to 2'b00, which generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.</p> <p>1'b1: The core drives the phy_opmode_o signal on the UTMI+ to 2'b01, which generates a device disconnect event to the USB host.</p> <p>Note: This bit can be also used for ULPI/FS Serial interfaces.  Note: This bit is not impacted by a soft reset.</p>	0x1
RmtWkUpSig	[0]	R/W	<p>Remote Wakeup Signaling (RmtWkUpSig)  When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must Set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1-15 ms after setting it.</p> <p>Remote Wakeup Signaling (RmtWkUpSig) When LPM is enabled, In L1 state the behavior of this bit is as follows:  When the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the Sleep state. As specified in the LPM specification, the hardware will automatically clear this bit after a time of 50 micro sec (TL1DevDrvResume) after set by application. Application should not set this bit when GLPMCFG.bRemoteWake from the previous LPM transaction was zero.</p>	0x0

### 7.4.43 DSTS

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x808 Reset Value = 0x2

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	RESERVED	-
DevLnSts	[23:22]	R	Device Line Status (DevLnSts) Indicates the current logic level USB data lines DevLnSts[1]: Logic level of D+ DevLnSts[0]: Logic level of D-	0x0
SOFFN	[21:8]	R	Frame or Microframe Number of the Received SOF (SOFFN) When the core is operating at high speed, this field contains a microframe number. When the core is operating at full or low speed, this field contains a Frame number. Note: This register may return a non zero value if read immediately after power on reset. In case the register bit reads non zero immediately after power on reset it does not indicate that SOF has been received from the host. The read value of this interrupt is valid only after a valid connection between host and device is established.	0x0
RSVD	[7:4]	-	RESERVED	-
ErrticErr	[3]	R	Erratic Error (ErrticErr) The core sets this bit to report any erratic errors (phy_rxvalid_i/phy_rxvlhd_i or phy_rxactive_i is asserted For at least 2 ms, due to PHY error) seen on the UTMI+. Because of erratic errors, DWC_otg goes into Suspended state and an interrupt is generated to the application with Early Suspend bit of the Core Interrupt register (GINTSTS.ErlySusp). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.	0x0
EnumSpd	[2:1]	R	Enumerated Speed (EnumSpd) Indicates the speed at which the DWC_otg core has come up after speed detection through a chirp sequence. 2'b00: High speed (PHY clock is running at 30 or 60 MHz) 2'b01: Full speed (PHY clock is running at 30 or 60 MHz) 2'b10: Low speed (PHY clock is running at 6 MHz) 2'b11: Full speed (PHY clock is running at 48 MHz) Low speed is not supported For devices using a UTMI+ PHY.	0x1
SuspSts	[0]	R	Suspend Status (SuspSts) In Device mode, this bit is Set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the phy_line_state_i signal For an extended period of time. The core comes out of the suspend: When there is any activity on the phy_line_state_i signal When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig).	0x0

### 7.4.44 DIEPMSK

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x810 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:14]	-	RESERVED	-
NAKMsk	[13]	R/W	NAK interrupt Mask (NAKMsk)	0x0
BNAInlntrMsk	[9]	R/W	BNA interrupt Mask (BNAInlntrMsk)	0x0
TxfifoUndrnMsk	[8]	R/W	Fifo Underrun Mask (TxfifoUndrnMsk)	0x0
RSVD	[7]	-	RESERVED	-

INEPNakEffMsk	[6]	R/W	IN Endpoint NAK Effective Mask (INEPNakEffMsk)	0x0
INTknEPMisMsk	[5]	R/W	IN Token received with EP Mismatch Mask (INTknEPMisMsk)	0x0
INTknTXFEmpMsk	[4]	R/W	IN Token Received When TxFIFO Empty Mask (INTknTXFEmpMsk)	0x0
TimeOUTMsk	[3]	R/W	Timeout Condition Mask (TimeOUTMsk) (Non-isochronous endpoints)	0x0
AHBErrMsk	[2]	R/W	AHB Error Mask (AHBErrMsk)	0x0
EPDisbldMsk	[1]	R/W	Endpoint Disabled Interrupt Mask (EPDisbldMsk)	0x0
XferComplMsk	[0]	R/W	Transfer Completed Interrupt Mask (XferComplMsk)	0x0

#### 7.4.45 DOEPMSK

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x814 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETMsk	[14]	R/W	NYET interrupt Mask (NYETMsk)	0x0
NAKMsk	[13]	R/W	NAK interrupt Mask (NAKMsk)	0x0
BbleErrMsk	[12]	R/W	Babble Error interrupt Mask (BbleErrMsk)	0x0
BnaOutIntrMsk	[9]	R/W	BNA interrupt Mask (BnaOutIntrMsk)	0x0
OutPktErrMsk	[8]	R/W	OUT Packet Error Mask (OutPktErrMsk)	0x0
RSVD	[7]	-	RESERVED	-
Back2BackSETup	[6]	R/W	Back-to-Back SETUP Packets Received Mask (Back2BackSETup) Applies to control OUT endpoints only.	0x0
StsPhseRcvdMsk	[5]	R/W	Status Phase Received Mask (StsPhseRcvdMsk) Applies to control OUT endpoints only.	0x0
OUTTknEPdisMsk	[4]	R/W	OUT Token Received when Endpoint Disabled Mask (OUTTknEPdisMsk) Applies to control OUT endpoints only.	0x0
SetUPMsk	[3]	R/W	SETUP Phase Done Mask (SetUPMsk) Applies to control endpoints only.	0x0
AHBErrMsk	[2]	R/W	AHB Error (AHBErrMsk)	0x0
EPDisbldMsk	[1]	R/W	Endpoint Disabled Interrupt Mask (EPDisbldMsk)	0x0
XferComplMsk	[0]	R/W	Transfer Completed Interrupt Mask (XferComplMsk)	0x0

#### 7.4.46 DAINT

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x818 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
OutEPInt15	[31]	R	OUT Endpoint 15 Interrupt Bit	0x0
OutEPInt14	[30]	R	OUT Endpoint 14 Interrupt Bit	0x0
OutEPInt13	[29]	R	OUT Endpoint 13 Interrupt Bit	0x0
OutEPInt12	[28]	R	OUT Endpoint 12 Interrupt Bit	0x0
OutEPInt11	[27]	R	OUT Endpoint 11 Interrupt Bit	0x0
OutEPInt10	[26]	R	OUT Endpoint 10 Interrupt Bit	0x0

OutEPInt9	[25]	R	OUT Endpoint 9 Interrupt Bit	0x0
OutEPInt8	[24]	R	OUT Endpoint 8 Interrupt Bit	0x0
OutEPInt7	[23]	R	OUT Endpoint 7 Interrupt Bit	0x0
OutEPInt6	[22]	R	OUT Endpoint 6 Interrupt Bit	0x0
OutEPInt5	[21]	R	OUT Endpoint 5 Interrupt Bit	0x0
OutEPInt4	[20]	R	OUT Endpoint 4 Interrupt Bit	0x0
OutEPInt3	[19]	R	OUT Endpoint 3 Interrupt Bit	0x0
OutEPInt2	[18]	R	OUT Endpoint 2 Interrupt Bit	0x0
OutEPInt1	[17]	R	OUT Endpoint 1 Interrupt Bit	0x0
OutEPInt0	[16]	R	OUT Endpoint 0 Interrupt Bit	0x0
InEplnt15	[15]	R	IN Endpoint 15 Interrupt Bit	0x0
InEplnt14	[14]	R	IN Endpoint 14 Interrupt Bit	0x0
InEplnt13	[13]	R	IN Endpoint 13 Interrupt Bit	0x0
InEplnt12	[12]	R	IN Endpoint 12 Interrupt Bit	0x0
InEplnt11	[11]	R	IN Endpoint 11 Interrupt Bit	0x0
InEplnt10	[10]	R	IN Endpoint 10 Interrupt Bit	0x0
InEplnt9	[9]	R	IN Endpoint 9 Interrupt Bit	0x0
InEplnt8	[8]	R	IN Endpoint 8 Interrupt Bit	0x0
InEplnt7	[7]	R	IN Endpoint 7 Interrupt Bit	0x0
InEplnt6	[6]	R	IN Endpoint 6 Interrupt Bit	0x0
InEplnt5	[5]	R	IN Endpoint 5 Interrupt Bit	0x0
InEplnt4	[4]	R	IN Endpoint 4 Interrupt Bit	0x0
InEplnt3	[3]	R	IN Endpoint 3 Interrupt Bit	0x0
InEplnt2	[2]	R	IN Endpoint 2 Interrupt Bit	0x0
InEplnt1	[1]	R	IN Endpoint 1 Interrupt Bit	0x0
InEplnt0	[0]	R	IN Endpoint 0 Interrupt Bit	0x0

#### 7.4.47 DAINTMSK

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x81C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
OutEPMsk15	[31]	R/W	OUT Endpoint 15 Interrupt mask Bit	0x0
OutEPMsk14	[30]	R/W	OUT Endpoint 14 Interrupt mask Bit	0x0
OutEPMsk13	[29]	R/W	OUT Endpoint 13 Interrupt mask Bit	0x0
OutEPMsk12	[28]	R/W	OUT Endpoint 12 Interrupt mask Bit	0x0
OutEPMsk11	[27]	R/W	OUT Endpoint 11 Interrupt mask Bit	0x0
OutEPMsk10	[26]	R/W	OUT Endpoint 10 Interrupt mask Bit	0x0
OutEPMsk9	[25]	R/W	OUT Endpoint 9 Interrupt mask Bit	0x0
OutEPMsk8	[24]	R/W	OUT Endpoint 8 Interrupt mask Bit	0x0
OutEPMsk7	[23]	R/W	OUT Endpoint 7 Interrupt mask Bit	0x0
OutEPMsk6	[22]	R/W	OUT Endpoint 6 Interrupt mask Bit	0x0
OutEPMsk5	[21]	R/W	OUT Endpoint 5 Interrupt mask Bit	0x0
OutEPMsk4	[20]	R/W	OUT Endpoint 4 Interrupt mask Bit	0x0
OutEPMsk3	[19]	R/W	OUT Endpoint 3 Interrupt mask Bit	0x0
OutEPMsk2	[18]	R/W	OUT Endpoint 2 Interrupt mask Bit	0x0
OutEPMsk1	[17]	R/W	OUT Endpoint 1 Interrupt mask Bit	0x0
OutEPMsk0	[16]	R/W	OUT Endpoint 0 Interrupt mask Bit	0x0
InEpMsk15	[15]	R/W	IN Endpoint 15 Interrupt mask Bit	0x0
InEpMsk14	[14]	R/W	IN Endpoint 14 Interrupt mask Bit	0x0
InEpMsk13	[13]	R/W	IN Endpoint 13 Interrupt mask Bit	0x0

InEpMsk12	[12]	R/W	IN Endpoint 12 Interrupt mask Bit	0x0
InEpMsk11	[11]	R/W	IN Endpoint 11 Interrupt mask Bit	0x0
InEpMsk10	[10]	R/W	IN Endpoint 10 Interrupt mask Bit	0x0
InEpMsk9	[9]	R/W	IN Endpoint 9 Interrupt mask Bit	0x0
InEpMsk8	[8]	R/W	IN Endpoint 8 Interrupt mask Bit	0x0
InEpMsk7	[7]	R/W	IN Endpoint 7 Interrupt mask Bit	0x0
InEpMsk6	[6]	R/W	IN Endpoint 6 Interrupt mask Bit	0x0
InEpMsk5	[5]	R/W	IN Endpoint 5 Interrupt mask Bit	0x0
InEpMsk4	[4]	R/W	IN Endpoint 4 Interrupt mask Bit	0x0
InEpMsk3	[3]	R/W	IN Endpoint 3 Interrupt mask Bit	0x0
InEpMsk2	[2]	R/W	IN Endpoint 2 Interrupt mask Bit	0x0
InEpMsk1	[1]	R/W	IN Endpoint 1 Interrupt mask Bit	0x0
InEpMsk0	[0]	R/W	IN Endpoint 0 Interrupt mask Bit	0x0

#### 7.4.48 DVBUSDIS

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x828 Reset Value = 0x17D7

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
DVBUSDis	[15:0]	R/W	Device VBUS Discharge Time (DVBUSDis) Specifies the VBUS discharge time after VBUS pulsing during SRP. This value equals: VBUS discharge time in PHY clocks / 1, 024 The value you use depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width). Depending on your VBUS load, this value can need adjustment.	0x17D7

#### 7.4.49 DVBUISPULSE

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x82C Reset Value = 0x5B8

Name	Bit	Type	Description	Reset Value
RSVD	[31:12]	-	RESERVED	-
DVBUISPulse	[11:0]	R/W	Device VBUS Pulsing Time (DVBUISPulse) Specifies the VBUS pulsing time during SRP. This value equals: VBUS pulsing time in PHY clocks / 1, 024 The value you use depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width).	0x5B8

### 7.4.50 DTHRCTL

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x830 Reset Value = 0xC100020

Name	Bit	Type	Description	Reset Value
RSVD	[31:28]	-	RESERVED	-
ArbPrkEn	[27]	R/W	Arbiter Parking Enable (ArbPrkEn) This bit controls internal DMA arbiter parking For IN endpoints. When thresholding is enabled and this bit is Set to one, Then the arbiter parks on the IN endpoint For which there is a token received on the USB. This is done to avoid getting into underrun conditions. By Default the parking is enabled.	0x1
RSVD	[26]	-	RESERVED	-
RxThrLen	[25:17]	R/W	Receive Threshold Length (RxThrLen) This field specifies Receive thresholding size in DWORDS. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight DWORDS. The recommended value For ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBstLen).	0x8
RxThrEn	[16]	R/W	Receive Threshold Enable (RxThrEn) When this bit is set, the core enables thresholding in the receive direction. Note: We recommends that you do not enable RxThrEn, because it may cause issues in the RxFIFO especially during error conditions such as RxError and Babble.	0x0
RSVD	[15:13]	-	RESERVED	-
AHBThrRatio	[12:11]	R/W	AHB Threshold Ratio (AHBThrRatio) These bits define the ratio between the AHB threshold and the MAC threshold for the transmit path only. The AHB threshold always remains less than or equal to the USB threshold, because this does not increase overhead. Both the AHB and the MAC threshold must be DWORD-aligned. The application needs to program TxThrLen and the AHBThrRatio to make the AHB Threshold value WORD aligned. If the AHB threshold value is not WORD aligned, the core might not behave correctly. When programming the TxThrLen and AHBThrRatio, the application must ensure that the minimum AHB threshold value does not go below 8 WORDS to meet the USB turnaround time requirements. 2'b00: AHB threshold = MAC threshold 2'b01: AHB threshold = MAC threshold / 2 2'b10: AHB threshold = MAC threshold / 4 2'b11: AHB threshold = MAC threshold / 8	0x0

TxThrLen	[10:2]	R/W	Transmit Threshold Length (TxThrLen) This field specifies Transmit thresholding size in DWORDS. This also forms the MAC threshold and specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmit on the USB. The threshold length has to be at least eight DWORDS when the value of AHBThrRatio is 2'h00. In case the AHBThrRatio is non zero the application needs to ensure that the AHB Threshold value does not go below the recommended eight DWORD. This field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAH-BCFG.HBstLen).	0x8
ISOThrEn	[1]	R/W	ISO IN Endpoints Threshold Enable. (ISOThrEn) When this bit is Set, the core enables thresholding For isochronous IN endpoints.	0x0
NonISOThrEn	[0]	R/W	Non-ISO IN Endpoints Threshold Enable. (NonISOThrEn) When this bit is Set, the core enables thresholding For Non Isochronous IN endpoints.	0x0

#### 7.4.51 DIEPEMPMSK

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x834 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
InEpTxfEmpMsk	[15:0]	R/W	IN EP Tx FIFO Empty Interrupt Mask Bits (InEpTxfEmpMsk) These bits acts as mask bits For DIEPINTn. TxFEmp interrupt One bit per IN Endpoint: Bit 0 For IN EP 0, bit 15 For IN EP 15	0x0

### 7.4.52 DIEPCTL0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x900 Reset Value = 0x8000

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	Endpoint Enable (EPEna) When Scatter/Gather DMA mode is enabled, For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. When Scatter/Gather DMA mode is disabled such as in buffer pointer based DMA mode this bit indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting the following interrupts on this endpoint: Endpoint Disabled Transfer Completed	0x0
EPDis	[30]	R/W	Endpoint Disable (EPDis) The application sets this bit to stop transmitting data on an endpoint, even before the transfer For that endpoint is complete. The application must wait For the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled Interrupt. The application must Set this bit only If Endpoint Enable is already Set For this endpoint.	0x0
RSVD	[29:28]	-	RESERVED	-
SNAK	[27]	W	Set NAK (SNAK) A write to this bit sets the NAK bit For the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.	0x0
CNAK	[26]	W	Clear NAK (CNAK) A write to this bit clears the NAK bit For the endpoint.	0x0
TxFNum	[25:22]	R/W	TxFIFO Number (TxFNum) For Shared FIFO operation, this value is always Set to 0, indicating that control IN endpoint 0 data is always written in the Non-Periodic Transmit FIFO. For Dedicated FIFO operation, this value is Set to the FIFO number that is assigned to IN Endpoint 0.	0x0
Stall	[21]	R/W	STALL Handshake (Stall) The application can only Set this bit, and the core clears it, when a SETUP token is received For this endpoint. If a NAK bit, Global Nonperiodic IN NAK, or Global OUT NAK is Set along with this bit, the STALL bit takes priority.	0x0
RSVD	[20]	-	RESERVED	-
EPType	[19:18]	R	Endpoint Type (EPType) Hardcoded to 00 for control.	0x0
NAKsts	[17]	R	NAK Status (NAKsts) Indicates the following: 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status 1'b1: The core is transmitting NAK handshakes on this endpoint. When this bit is Set, either by the application or core, the core stops transmitting data, even If there is data available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.	0x0

RSVD	[16]	-	RESERVED	-
USBActEP	[15]	R	USB Active Endpoint (USBActEP) This bit is always SET to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.	0x1
RSVD	[10:2]	-	RESERVED	-
MPS	[1:0]	R/W	Maximum Packet Size (MPS) Applies to IN and OUT endpoints. The application must program this field with the maximum packet size For the current logical endpoint. 2'b00: 64 bytes 2'b01: 32 bytes 2'b10: 16 bytes 2'b11: 8 bytes	0x0

#### 7.4.53 DIEPINT0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x908 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETInrpt	[14]	R/W	NYET Interrupt (NYETInrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKInrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only The core generates this interrupt when it detects a transmit FIFO underrun condition in threshold mode For this endpoint.	0x0
TxFEemp	[7]	R	Transmit FIFO Empty (TxFEemp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEempLvl)).	0x1

INEPNakEff	[6]	R/W	<p>IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK.</p> <p>This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.</p> <p>This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.</p>	0x0
INTknEPMis	[5]	R/W	<p>IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.</p>	0x0
INTknTXFEmp	[4]	R/W	<p>IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.</p>	0x0
TimeOUT	[3]	R/W	<p>Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.</p>	0x0
AHBErr	[2]	R/W	<p>AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.</p>	0x0
EPDisbld	[1]	R/W	<p>Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.</p>	0x0
XferCompl	[0]	R/W	<p>Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.</p>	0x0

#### 7.4.54 DIEPTSIZ0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x910 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:21]	-	RESERVED	-
PktCnt	[20:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. In Endpoints : This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO. OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.	0x0
RSVD	[18:7]	-	RESERVED	-
XferSize	[6:0]	R/W	Transfer Size (XferSize) This field contains the transfer size in bytes for the current endpoint. The transfer size (XferSize) = Sum of buffer sizes across all descriptors in the list for the endpoint. In Buffer DMA, the core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. IN Endpoints: The core decrements this field every time a packet from the external memory is written to the TxFIFO. OUT Endpoints: The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.55 DIEPDMA0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x914 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.56 DTXFSTS0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x918 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail) Indicates the amount of free space available in the Endpoint Tx FIFO. Values are in terms of 32-bit words. 16'h0: Endpoint Tx FIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 < n < 32,768) 16'h8000: 32,768 words available Others: Reserved	0x0

#### 7.4.57 DIEPDMAB0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x91C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.58 DIEPCTL1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x920 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)          Applies to interrupt/bulk IN and OUT endpoints only.          Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO          In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)          Applies to isochronous IN and OUT endpoints only.          Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.          When Scatter/Gather DMA mode is enabled, this field is reserved.          The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)          A write to this bit sets the NAK bit For the endpoint.          Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)          A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)          Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO          Others: Specified Periodic TxFIFO.number          Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.          This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)          Applies to non-control, non-isochronous IN and OUT endpoints only.          The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W          Applies to control endpoints only.          The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)          This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control          2'b01: Isochronous          2'b10: Bulk          2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.59 DIEPINT1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x928 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.60 DIEPTSIZ1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x930 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.61 DIEPDMA1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x934 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.62 DTXFSTS1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x938 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.63 DIEPDMA1B

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x93C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.64 DIEPCTL2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x940 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.65 DIEPINT2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x948 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.66 DIEPTSIZ2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x950 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.67 DIEPDMA2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x954 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.68 DTXFSTS2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x958 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.69 DIEPDMA2B

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x95C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.70 DIEPCTL3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x960 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.71 DIEPINT3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x968 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.72 DIEPTSI23

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x970 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

### 7.4.73 DIEPDMA3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x974 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

### 7.4.74 DTXFSTS3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x978 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

### 7.4.75 DIEPDMA3B

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x97C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.76 DIEPCTL4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x980 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.77 DIEPINT4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x988 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.78 DIEPTSI24

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x990 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.79 DIEPDMA4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x994 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.80 DTXFSTS4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x998 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.81 DIEPDMA8B4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x99C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.82 DIEPCTL5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9A0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.83 DIEPINT5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9A8 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.84 DIEPTSI25

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9B0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.85 DIEPDMA5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9B4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.86 DTXFSTS5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9B8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.87 DIEPDMA5B

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9BC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.88 DIEPCTL6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9C0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.89 DIEPINT6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9C8 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.90 DIEPTISZ6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9D0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

### 7.4.91 DIEPDMA6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9D4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

### 7.4.92 DTXFSTS6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9D8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

### 7.4.93 DIEPDMA8B

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9DC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.94 DIEPCTL7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9E0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.95 DIEPINT7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9E8 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.96 DIEPTSI7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9F0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.97 DIEPDMA7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9F4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.98 DTXFSTS7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9F8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.99 DIEPDMA8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0x9FC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.100 DIEPCTL8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA00 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)          Applies to interrupt/bulk IN and OUT endpoints only.          Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO          In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)          Applies to isochronous IN and OUT endpoints only.          Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.          When Scatter/Gather DMA mode is enabled, this field is reserved.          The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)          A write to this bit sets the NAK bit For the endpoint.          Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)          A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)          Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO          Others: Specified Periodic TxFIFO.number          Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.          This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)          Applies to non-control, non-isochronous IN and OUT endpoints only.          The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W          Applies to control endpoints only.          The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)          This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control          2'b01: Isochronous          2'b10: Bulk          2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.101 DIEPINT8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA08 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.102 DIEPTSIZ8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA10 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.103 DIEPDMA8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA14 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.104 DTXFSTS8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA18 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.105 DIEPDMA8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA1C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.106 DIEPCTL9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA20 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.107 DIEPINT9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA28 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.108 DIEPTSIZ9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA30 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.109 DIEPDMA9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA34 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.110 DTXFSTS9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA38 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.111 DIEPDMA9B

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA3C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.112 DIEPCTL10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA40 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)          Applies to interrupt/bulk IN and OUT endpoints only.          Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO          In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)          Applies to isochronous IN and OUT endpoints only.          Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.          When Scatter/Gather DMA mode is enabled, this field is reserved.          The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)          A write to this bit sets the NAK bit For the endpoint.          Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)          A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)          Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO          Others: Specified Periodic TxFIFO.number          Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.          This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)          Applies to non-control, non-isochronous IN and OUT endpoints only.          The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W          Applies to control endpoints only.          The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)          This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control          2'b01: Isochronous          2'b10: Bulk          2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.113 DIEPINT10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA48 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.114 DIEPTSIZ10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA50 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.115 DIEPDMA10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA54 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.116 DTXFSTS10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA58 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.117 DIEPDMA10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA5C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.118 DIEPCTL11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA60 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.119 DIEPINT11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA68 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.120 DIEPTSIZ11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA70 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.121 DIEPDMA11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA74 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.122 DTXFSTS11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA78 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.123 DIEPDMA11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA7C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.124 DIEPCTL12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA80 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.125 DIEPINT12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA88 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.126 DIEPTSIZ12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA90 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.127 DIEPDMA12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA94 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.128 DTXFSTS12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA98 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.129 DIEPDMA12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xA9C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.130 DIEPCTL13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAA0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)          Applies to interrupt/bulk IN and OUT endpoints only.          Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO          In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)          Applies to isochronous IN and OUT endpoints only.          Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.          When Scatter/Gather DMA mode is enabled, this field is reserved.          The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)          A write to this bit sets the NAK bit For the endpoint.          Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)          A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)          Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO          Others: Specified Periodic TxFIFO.number          Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.          This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)          Applies to non-control, non-isochronous IN and OUT endpoints only.          The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W          Applies to control endpoints only.          The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)          This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control          2'b01: Isochronous          2'b10: Bulk          2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.131 DIEPINT13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAA8 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.132 DIEPTSIZ13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAB0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.133 DIEPDMA13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAB4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.134 DTXFSTS13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAB8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.135 DIEPDMA13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xABC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.136 DIEPCTL14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAC0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO</p> <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.137 DIEPINT14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAC8 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.138 DIEPTSIZ14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAD0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.139 DIEPDMA14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAD4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.140 DTXFSTS14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAD8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.141 DIEPDMA14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xADC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.142 DIEPCTL15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAE0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.            This field is not applicable for Scatter/Gather DMA mode.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.      When Scatter/Gather DMA mode is enabled, this field is reserved.      The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
TxFNum	[25:22]	R/W	<p>TxFIFO Number (TxFNum)      Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <p>4'h0: Non-Periodic TxFIFO      Others: Specified Periodic TxFIFO.number      Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p>Dedicated FIFO Operationthese bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.      This field is valid only for IN endpoints.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0

NAKsts	[17]	R	<p>NAK Status (NAKsts)</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
DPID	[16]	R	<p>Endpoint Data PID (DPID)</p> <p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.143 DIEPINT15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAE8 Reset Value = 0x80

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	RESERVED	-
NYETIntrpt	[14]	R/W	NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.	0x0
NAKIntrpt	[13]	R/W	NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.	0x0
BbleErr	[12]	R/W	NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.	0x0
PktDrpSts	[11]	R/W	Packet Drop Status (PktDrpSts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.	0x0
BNAIntr	[9]	R/W	BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done	0x0
TxfifoUndrn	[8]	R/W	Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition For this endpoint.	0x0
TxFEmp	[7]	R	Transmit FIFO Empty (TxFEmp) This bit is valid only For IN Endpoints This interrupt is asserted when the TxFIFO For this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).	0x1
INEPNakEff	[6]	R/W	IN Endpoint NAK Effective (INEPNakEff) Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	0x0
INTknEPMis	[5]	R/W	IN Token Received with EP Mismatch (INTknEPMis) Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one For which the IN token was received. This interrupt is asserted on the endpoint For which the IN token was received.	0x0
INTknTXFEmp	[4]	R/W	IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint For which the IN token was received.	0x0

TimeOUT	[3]	R/W	Timeout Condition (TimeOUT) In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB For the last IN token on this endpoint.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, For this endpoint.	0x0

#### 7.4.144 DIEPTSIZ15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAF0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
MC	[30:29]	R/W	Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)	0x0
PktCnt	[28:19]	R/W	Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data For endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	0x0

#### 7.4.145 DIEPDMA15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAF4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.146 DTXFSTS15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAF8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	RESERVED	-
INEPTxFSpcAvail	[15:0]	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpcAvail)</p> <p>Indicates the amount of free space available in the Endpoint Tx FIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>16'h0: Endpoint Tx FIFO is full</p> <p>16'h1: 1 word available</p> <p>16'h2: 2 words available</p> <p>16'hn: n words available (where 0 &lt; n &lt; 32,768)</p> <p>16'h8000: 32,768 words available</p> <p>Others: Reserved</p>	0x0

#### 7.4.147 DIEPDMA15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xAFC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

#### 7.4.148 DOEPCTL0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB00 Reset Value = 0x8000

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            When Scatter/Gather DMA mode is enabled, For OUT endpoints this bit indicates that the descriptor structure and data buffer to receive data is setup.            When Scatter/Gather DMA mode is disabled(such as For buffer-pointer based DMA mode) this bit indicates that the application has allocated the memory to start receiving data from the USB.</p> <p>The core clears this bit before setting any of the following interrupts on this endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: In DMA mode, this bit must be Set For the core to transfer SETUP data packets into memory.</p>	0x0
EPDis	[30]	R	Endpoint Disable (EPDis) The application cannot disable control OUT endpoint 0.	0x0
RSVD	[29:28]	-	RESERVED	-
SNAK	[27]	W	<p>Set NAK (SNAK)            A write to this bit sets the NAK bit For the endpoint.            Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)            A write to this bit clears the NAK bit For the endpoint.</p>	0x0
RSVD	[25:22]	-	RESERVED	-
Stall	[21]	R/W	<p>STALL Handshake (Stall)            The application can only Set this bit, and the core clears it, when a SETUP token is received For this endpoint. If a NAK bit or Global OUT NAK is Set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)            This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R	<p>Endpoint Type (EPType)            Hardcoded to 2'b00 For control.</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)            Indicates the following:            1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.            1'b1: The core is transmitting NAK handshakes on this endpoint.            When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
RSVD	[16]	-	RESERVED	-
USBActEP	[15]	R	<p>USB Active Endpoint (USBActEP)            This bit is always Set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.</p>	0x1
RSVD	[14:2]	-	RESERVED	-
MPS	[1:0]	R	<p>Maximum Packet Size (MPS)            The maximum packet size For control OUT endpoint 0 is the same as what is programmed in control IN Endpoint 0.            2'b00: 64 bytes            2'b01: 32 bytes            2'b10: 16 bytes            2'b11: 8 bytes</p>	0x0

### 7.4.149 DOEPINT0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB08 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.150 DOEPTSIZ0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB10 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
SUPCnt	[30:29]	R/W	SETUP Packet Count (SUPCnt) This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0
RSVD	[28:20]	-	RESERVED	-
PktCnt	[19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
RSVD	[18:7]	-	RESERVED	-

XferSize	[6:0]	R/W	<p>Transfer Size (XferSize)            Indicates the transfer size in bytes For endpoint 0. The core            interrupts the application only after it has exhausted the transfer            size amount of data. The transfer size can be Set to the            maximum packet size of the endpoint, to be interrupted at the            end of each packet.            The core decrements this field every time a packet is read from            the RxFIFO and written to the external memory.</p>	0x0
----------	-------	-----	--	-----

#### 7.4.151 DOEPDMA0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB14 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.            Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.            This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.            When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.            When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p>	0x0

#### 7.4.152 DOEPDMAB0

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB1C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.            This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p>	0x0

### 7.4.153 DOEPCTL1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB20 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1          This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.          1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.          1'b0: Even (micro)frame          1'b1: Odd (micro)frame          When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

#### 7.4.154 DOEPINT1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB28 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.155 DOEPTSIZ1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB30 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.156 DOEPDMA1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB34 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.157 DOEPDMAB1

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB3C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.158 DOEPCTL2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB40 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.159 DOEPINT2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB48 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.160 DOEPTSIZ2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB50 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.161 DOEPDMA2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB54 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.162 DOEPDMAB2

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB5C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.163 DOEPCTL3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB60 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1          This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.          1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.          1'b0: Even (micro)frame          1'b1: Odd (micro)frame          When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

#### 7.4.164 DOEPINT3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB68 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.165 DOEPTSIZ3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB70 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.166 DOEPDMA3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB74 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.167 DOEPDMAB3

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB7C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

#### 7.4.168 DOEPCTL4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB80 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.169 DOEPINT4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB88 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.170 DOEPTSIZ4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB90 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.171 DOEPDMA4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB94 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.172 DOEPDMAB4

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xB9C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.173 DOEPCTL5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBA0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.174 DOEPINT5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBA8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.175 DOEPTSIZ5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBB0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.176 DOEPDMA5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBB4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.177 DOEPDMAB5

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBCB Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.178 DOEPCTL6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBC0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.179 DOEPINT6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBC8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.180 DOEPTSI26

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBD0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.181 DOEPDMA6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBD4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.182 DOEPDMAB6

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBCD Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.183 DOEPCTL7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBE0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

#### 7.4.184 DOEPINT7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBE8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.185 DOEPTSI7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBF0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.186 DOEPDMA7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBF4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.187 DOEPDMAB7

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xBFC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

#### 7.4.188 DOEPCTL8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC00 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.189 DOEPINT8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC08 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.190 DOEPTSIZ8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC10 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.191 DOEPDMA8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC14 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.192 DOEPDMAB8

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC1C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.193 DOEPCTL9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC20 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

#### 7.4.194 DOEPINT9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC28 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.195 DOEPTSIZ9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC30 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.196 DOEPDMA9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC34 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.197 DOEPDMAB9

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC3C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.198 DOEPCTL10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC40 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.199 DOEPINT10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC48 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.200 DOEPTSIZ10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC50 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.201 DOEPDMA10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC54 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.202 DOEPDMAB10

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC5C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.203 DOEPCTL11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC60 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.204 DOEPINT11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC68 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.205 DOEPTSIZ11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC70 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.206 DOEPDMA11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC74 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.207 DOEPDMAB11

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC7C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.208 DOEPCTL12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC80 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.209 DOEPINT12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC88 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.210 DOEPTSIZ12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC90 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.211 DOEPDMA12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC94 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.212 DOEPDMAB12

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xC9C Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.213 DOEPCTL13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCA0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.214 DOEPINT13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCA8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.215 DOEPTSIZ13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCB0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.216 DOEPDMA13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCB4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.217 DOEPDMAB13

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCBC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.218 DOEPCTL14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCC0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.219 DOEPINT14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCC8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.220 DOEPTSIZ14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCD0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.221 DOEPDMA14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCD4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.222 DOEPDMAB14

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCDC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

### 7.4.223 DOEPCTL15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCE0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
EPEna	[31]	R/W	<p>Endpoint Enable (EPEna)            Applies to IN and OUT endpoints.            When Scatter/Gather DMA mode is enabled,            For IN endpoints this bit indicates that the descriptor structure and            data buffer with            data ready to transmit is setup.            For OUT endpoint it indicates that the descriptor structure and            data buffer to            receive data is setup.            When Scatter/Gather DMA mode is enabled such as for buffer-            pointer based            DMA mode:            - For IN endpoints, this bit indicates that data is ready to be trans-            mitted on the            endpoint.            - For OUT endpoints, this bit indicates that the application has al-            located the            memory to start receiving data from the USB.            - The core clears this bit before setting any of the following inter-            rupts on this            endpoint:            SETUP Phase Done            Endpoint Disabled            Transfer Completed            Note: For control endpoints in DMA mode, this bit must be set to            be able to transfer            SETUP data packets in memory.</p>	0x0
EPDis	[30]	R/W	<p>Endpoint Disable (EPDis)            Applies to IN and OUT endpoints.            The application sets this bit to stop transmitting/receiving data on            an endpoint, even            before the transfer for that endpoint is complete. The application            must wait for the            Endpoint Disabled interrupt before treating the endpoint as dis-            abled. The core clears            this bit before setting the Endpoint Disabled interrupt. The appli-            cation must set this bit            only if Endpoint Enable is already set for this endpoint.</p>	0x0
SetD1PID	[29]	W	<p>Set DATA1 PID (SetD1PID)            Applies to interrupt/bulk IN and OUT endpoints only.            Writing to this field sets the Endpoint Data PID (DPID) field in this            register to DATA1.            This field is applicable both for Scatter/Gather DMA mode and            non-Scatter/Gather            DMA mode.            Set Odd (micro)frame (SetOddFr)            Applies to isochronous IN and OUT endpoints only.            Writing to this field sets the Even/Odd (micro)frame (EO_FrNum)            field to odd            (micro)frame.</p>	0x0

SetD0PID	[28]	W	<p>Set DATA0 PID (SetD0PID)      Applies to interrupt/bulk IN and OUT endpoints only.      Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 WO      In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr)      Applies to isochronous IN and OUT endpoints only.      Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>	0x0
SNAK	[27]	W	<p>Set NAK (SNAK)      A write to this bit sets the NAK bit For the endpoint.      Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.</p>	0x0
CNAK	[26]	W	<p>Clear NAK (CNAK)      A write to this bit clears the NAK bit For the endpoint.</p>	0x0
Stall	[21]	R/W	<p>STALL Handshake (Stall)      Applies to non-control, non-isochronous IN and OUT endpoints only.      The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>1'b0 R_W      Applies to control endpoints only.      The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0
Snp	[20]	R/W	<p>Snoop Mode (Snp)      Applies to OUT endpoints only.      This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>	0x0
EPType	[19:18]	R/W	<p>Endpoint Type (EPType)      This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control      2'b01: Isochronous      2'b10: Bulk      2'b11: Interrupt</p>	0x0
NAKsts	[17]	R	<p>NAK Status (NAKsts)      Indicates the following:      1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.      1'b1: The core is transmitting NAK handshakes on this endpoint.      When either the application or the core sets this bit:      The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.      For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.      For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.      Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	0x0

DPID	[16]	R	<p>Endpoint Data PID (DPID)          Applies to interrupt/bulk IN and OUT endpoints only.          Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0          1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>1'b0 RO          Even/Odd (Micro)Frame (EO_FrNum)          In non-Scatter/Gather DMA mode:          Applies to isochronous IN and OUT endpoints only.          Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame          1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>	0x0
USBActEP	[15]	R/W	<p>USB Active Endpoint (USBActEP)          Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	0x0
MPS	[10:0]	R/W	<p>Maximum Packet Size (MPS)          The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	0x0

### 7.4.224 DOEPINT15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCE8 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
StupPktRcvd	[15]	R/W	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <p>1'b0: No Setup packet received 1'b1: Setup packet received Reset: 1'b0</p>	0x0
NYETIntrpt	[14]	R/W	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p>	0x0
NAKIntrpt	[13]	R/W	<p>NAK Interrupt (NAKInterrupt) The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.</p>	0x0
BbleErr	[12]	R/W	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p>	0x0
PktDrpsts	[11]	R/W	<p>Packet Drop Status (PktDrpsts) This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p>	0x0
BNAIntr	[9]	R/W	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready For the Core to process, such as Host busy or DMA done</p>	0x0
OutPktErr	[8]	R/W	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error For non-Isochronous OUT packet.</p>	0x0
Back2BackSETup	[6]	R/W	<p>Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets For this particular endpoint. For information about handling this interrupt,</p>	0x0

StsPhseRcvd	[5]	R/W	Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only For Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.	0x0
OUTTknEPdis	[4]	R/W	OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint For which the OUT token was received.	0x0
setUp	[3]	R/W	SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase For the control endpoint is complete and no more back-to-back SETUP packets were received For the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	0x0
AHBErr	[2]	R/W	AHB Error (AHBErr) Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	0x0
EPDisbld	[1]	R/W	Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	0x0
XferCompl	[0]	R/W	Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit For the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB. For this endpoint.	0x0

#### 7.4.225 DOEPTSIZ15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCF0 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	RESERVED	-
RxDPID	[30:29]	R	Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA SETUP Packet Count (SUPCnt) Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets	0x0

PktCnt	[28:19]	R/W	Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.	0x0
XferSize	[18:0]	R/W	Transfer Size (XferSize) Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	0x0

#### 7.4.226 DOEPDMA15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCF4 Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMAAddr	[31:0]	R/W	Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	0x0

#### 7.4.227 DOEPDMAB15

- Base Address: 20D0\_0000(USB20OTG)
- Address = Base Address + 0xCFC Reset Value = 0x0

Name	Bit	Type	Description	Reset Value
DMABufferAddr	[31:0]	R	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	0x0

# 8 DDRC

## 8.1 PIN Description

### 8.1.1 AXI Interfaces

Name	Direction	Width	Description
axi_clk	input	1	AXI CLK
axi_areset	input	1	AXI RESET
axi_arid	input	7	AXI RID
axi_araddr	input	32	AXI RADDR
axi_arlen	input	4	AXI RLEN
axi_arsize	input	3	AXI RSIZE
axi_arburst	input	2	AXI RBURST
axi_arlock	input	2	AXI RLOCK
axi_arcache	input	4	AXI RCACHE
axi_arprot	input	3	AXI RPROT
axi_arvalid	input	1	AXI RVALID
axi_arready	output	1	AXI RREADY
axi_rid	output	7	AXI ID
axi_rdata	output	64	AXI DATA
axi_resp	output	2	AXI RESP
axi_rlast	output	1	AXI LAST
axi_rvalid	output	1	AXI VALID
axi_rready	input	1	AXI READY
axi_awid	input	7	AXI WID
axi_awaddr	input	32	AXI WADDR
axi_awlen	input	4	AXI WLEN
axi_awsize	input	3	AXI WSIZE
axi_awburst	input	2	AXI WBURST
axi_awlock	input	2	AXI WLOCK
axi_awcache	input	4	AXI WCACHE
axi_awprot	input	3	AXI WPROT
axi_awvalid	input	1	AXI WVALID
axi_awready	output	1	AXI WREADY
axi_wid	input	7	AXI ID
axi_wdata	input	64	AXI DATA
axi_wstrb	input	8	AXI STRB
axi_wlast	input	1	AXI LAST
axi_wvalid	input	1	AXI VALID
axi_wready	output	1	AXI READY
axi_bid	output	7	AXI ID
axi_bresp	output	2	AXI RESP
axi_bvalid	output	1	AXI VALID
axi_bready	input	1	AXI READY

### 8.1.2 APB Interfaces

Name	Direction	Width	Description
PCLKEN	input	1	PCLK Enable
PADDR	input	8	PADDR
PSEL	input	1	PSEL
PENABLE	input	1	PENABLE
PWRITE	input	1	PWRITE
PWDATA	input	32	PWDATA
PRDATA	output	1	PRDATA
PREADY	output	32	PREADY
PSLVERR	output	1	PSLVERR

### 8.1.3 DDR Interfaces

Name	Direction	Width	Description
ddr_clk200MHz	input	1	only use FPGA Synthesis :200MHz. otherwise, connect to GND
ddr_clk0	input	1	DDR CLK
ddr_clk90	input	1	DDR CLK Phase 90 shift
ddr_reset	input	1	DDR RESET
ddr_cfg_odten	output	1	DDR On-Die-Termination [ODT] Enable [odt_en,odtsel] : 2'b00 : No Termination 2'b01 : 150 ohm 2'b10 : 75 ohm 2'b11 : 50 ohm
ddr_cfg_odtsel	output	1	refer to ddr_cfg_odten signal.
ddr_cfg_drivesel	output	1	DDR Drive 1'b0 : 60% Drive[SSTL I] 1'b1 : Full Drive [SSTL II]
ddr_cfg_strength	output	2	DDR Drive Strength Selection 2'b00 : One-Eight Drive 2'b01 : Quarter Drive 2'b10 : Half Drive 2'b11 : Full Drive
ddr_cfg_mddrsel	output	1	DDR Mode Sel
ddr_ck	output	1	DDR Clock
ddr_dq_t	output	pDataBits(32)	DDR DQ Direction Active-Low
ddr_dq_o	output	pDataBits(32)	DDR DQ Output
ddr_dq_i	input	pDataBits(32)	DDR DQ Input
ddr_dqs_t	output	pDataBits/8(4)	DDR DQS Direction Active-Low
ddr_dqs_o	output	pDataBits/8(4)	DDR DQS Output
ddr_dqs_i	input	pDataBits/8(4)	DDR DQS Input
ddr_dqm	output	pDataBits/8(4)	DDR DQM
ddr_odt	output	1	DDR ODT for DDR2 PAD Only
ddr_addr	output	pRowaBits(14)	DDR Address
ddr_ba	output	2	DDR Bank
ddr_cke	output	1	DDR Clock Enable
ddr_cs_n	output	1	DDR Chip Select
ddr_ras_n	output	1	DDR Row Address Strobe
ddr_cas_n	output	1	DDR Column Address Strobe
ddr_we_n	output	1	DDR Write Enable

### 8.1.4 Configuration Done interface

Name	Direction	Width	Description
ddr_initdone	output	1	Initialize Done
ddr_initkey	output	1	Initialize Success

## 8.2 DDRC Register Description

### 8.2.1 Register Summary

Offset	Register	Reset Value	Type	Description
0x00	DDR_PHY_CONFIG	0x00000000	R/W	DDR PHY initialization start and status
0x04	DDR_ADDR_SIZE	0x00000001	R/W	DDR Address Size Configuration[Bank, Row, Column]
0x08	DDR_TIMING_0	0x000C0C06	R/W	DDR Timing Configuration 0
0x0C	DDR_TIMING_1	0x00623233	R/W	DDR Timing Configuration 1
0x10	DDR_TIMING_2	0x753008B3	R/W	DDR Timing Configuration 2
0x14	DDR_LMR_EXT_STD	0x00000000	R/W	DDR Standard and Extended Load Mode Registers
0x18	DDR_LMR_EXT_3_2	0x00000000	R/W	DDR Extended2 and Extended3 Load Mode Registers
0x1C	DDR_CFG_RDQ_LATENCY	0x00000001	R	DDR Read DQ Align and Latency Configuration
0x20	DDR_CFG_RDQ_DLYVAL	0x00000000	R/W	DDR Read DQ Delay Cell Value Configuration
0x24	DDR_CFG_CMD_DLYVAL	0x00000000	R/W	DDR Write DQ, Clock, Command and Address Delay Cell Value Configuration
0x40	DDR_STS_RDQ_LATENCY	0x00000001	R	DDR Read DQ Align and Latency Status
0x48	DDR_STS_RDQ_DLYVAL	0x00000000	R	DDR Read DQ Delay Cell Value Status
0x4C	DDR_STS_CMD_DLYVAL	0x00000000	R	DDR Write DQ, Clock, Command and Address Delay Cell Value Status
0x50	DDR_STS_VERF_ERRPOS	0x00000000	R	DDR Auto Calibration Verify Error Status
0x54	DDR_STS_VERF_ERRDATA_L	0x00000000	R	DDR Auto Calibration Verify Error Data
0x58	DDR_STS_VERF_ERRDATA_H	0x00000000	R	DDR Auto Calibration Verify Error Data
0x60	DDR_STS_RDQCPT_R_BEAT_0_7	0x33333333	R	Auto Calibration : DDR Read Rise DQ Beat Capture Status[0-7]
0x64	DDR_STS_RDQCPT_R_BEAT_8_15	0x33333333	R	Auto Calibration : DDR Read Rise DQ Beat Capture Status[8-15]
0x68	DDR_STS_RDQCPT_R_BEAT_16_23	0x33333333	R	Auto Calibration : DDR Read Rise DQ Beat Capture Status[16-23]
0x6C	DDR_STS_RDQCPT_R_BEAT_24_31	0x33333333	R	Auto Calibration : DDR Read Rise DQ Beat Capture Status[24-31]
0x70	DDR_STS_RDQCPT_F_BEAT_0_7	0x33333333	R	Auto Calibration : DDR Read Fall DQ Beat Capture Status[0-7]
0x74	DDR_STS_RDQCPT_F_BEAT_8_15	0x33333333	R	Auto Calibration : DDR Read Fall DQ Beat Capture Status[8-15]
0x78	DDR_STS_RDQCPT_F_BEAT_16_23	0x33333333	R	Auto Calibration : DDR Read Fall DQ Beat Capture Status[16-23]
0x7C	DDR_STS_RDQCPT_F_BEAT_24_31	0x33333333	R	Auto Calibration : DDR Read Fall DQ Beat Capture Status[24-31]
0x80	DDR_STS_RDQCPT_BEAT_0_7	0x33333333	R	Auto Calibration : DDR Read DQ Beat Capture Status[0-7]
0x84	DDR_STS_RDQCPT_BEAT_8_15	0x33333333	R	Auto Calibration : DDR Read DQ Beat Capture Status[8-15]
0x88	DDR_STS_RDQCPT_BEAT_16_23	0x33333333	R	Auto Calibration : DDR Read DQ Beat Capture Status[16-23]
0x8C	DDR_STS_RDQCPT_BEAT_24_31	0x33333333	R	Auto Calibration : DDR Read DQ Beat Capture Status[24-31]
0x90	DDR_STS_RDQCPT_BYTE0_0_7	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane0 Capture Status[0-7]
0x94	DDR_STS_RDQCPT_BYTE0_8_15	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane0 Capture Status[8-15]
0x98	DDR_STS_RDQCPT_BYTE0_16_23	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane0 Capture Status[16-23]
0x9C	DDR_STS_RDQCPT_BYTE0_24_31	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane0 Capture Status[24-31]
0xA0	DDR_STS_RDQCPT_BYTE1_0_7	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane1 Capture Status[0-7]
0xA4	DDR_STS_RDQCPT_BYTE1_8_15	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane1 Capture Status[8-15]
0xA8	DDR_STS_RDQCPT_BYTE1_16_23	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane1 Capture Status[16-23]
0xAC	DDR_STS_RDQCPT_BYTE1_24_31	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane1 Capture Status[24-31]
0xB0	DDR_STS_RDQCPT_BYTE2_0_7	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane2 Capture Status[0-7]
0xB4	DDR_STS_RDQCPT_BYTE2_8_15	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane2 Capture Status[8-15]
0xB8	DDR_STS_RDQCPT_BYTE2_16_23	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane2 Capture Status[16-23]
0xBC	DDR_STS_RDQCPT_BYTE2_24_31	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane2 Capture Status[24-31]
0xC0	DDR_STS_RDQCPT_BYTE3_0_7	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane3 Capture Status[0-7]
0xC4	DDR_STS_RDQCPT_BYTE3_8_15	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane3 Capture Status[8-15]
0xC8	DDR_STS_RDQCPT_BYTE3_16_23	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane3 Capture Status[16-23]
0xCC	DDR_STS_RDQCPT_BYTE3_24_31	0x33333333	R	Auto Calibration : DDR Read DQ Byte Lane3 Capture Status[24-31]

## 8.2.2 DDR\_PHY\_CONFIG

Register	Bits	Type	Description
Start	[0]	W	PHY Initialization Start bit. This bit is automatically cleared to '0'.
Clear	[1]	W	PHY Initialization Sends the state machine to IDLE state, and starts the PHY initialization when the start signal is given again. This bit is automatically cleared to '0'.
Complete	[4]	R	PHY Initialization Complete. DDR can not be accessed until this bit is set to '1'.
Success	[5]	R	It is a bit to inform PHY Initialization Success. This bit is valid when the Complete bit is '1'. 0x0 : Fail 0x1 : Success
dis_calib	[7]	R/W	PHY Initialization is a bit that disables Auto Calibration. If this bit is '1', only DDR Initialization Power Sequence is executed and Auto Calibration is not performed.
dis_pwrseq	[8]	R/W	This bit disables Power Sequence Initialization. When this bit is set to '1', Power Sequence Initialization is not executed when Auto Calibration is executed.
dis_rdqcpt	[9]	R/W	Read DQ This bit disables DQ Capture execution. When this bit is set to '1', Read DQ Capture is not executed when Auto Calibration is executed.
dis_align	[10]	R/W	Read DQ This bit disables Align execution. When this bit is set to '1', Read DQ Align is not executed when executing Auto Calibration.
dis_latency	[11]	R/W	Read DQ This bit disables Latency execution. When this bit is set to '1', Read DQ Latency is not executed when Auto Calibration is executed.
dis_verify	[12]	R/W	Read DQ This is a bit that disables Verify execution. When this bit is set to '1', Read DQ Verify is not executed when Auto Calibration is executed.
err_ignore	[15]	R/W	Read DQ Align, Latency, Verify This is a bit to stop or ignore when an error occurs. If this bit is '0', it stops when an error occurs. If it is '1', it continues even if an error occurs.

err_ignore	error occur Calibration	error occur DDR Access
0x0	STOP	NO
0x1	Continue	YES

### 8.2.3 DDR\_ADDR\_SIZE

Register	Bits	Type	Description
col_size	[1:0]	R/W	It is a register that determines the Column Size of DDR / DDR2. The value of this register plus 8 of the set value (8 + col_size) is applied.
row_size	[3:2]	R/W	It is the register that determines the row size of DDR / DDR2. The value of this register plus 12 of the set value (12 + row_size) is applied.
bank_size	[4]	R/W	It is a register that determines the bank size of DDR / DDR2. The value of this register plus 2 of the set value (2 + bank_size) is applied.
addr_map	[5]	R/W	If it is '0', bank-row-col, and if it is '1', row-bank-col.
sel_mddr	[7]	R/W	It is the bit that selects DDR and DDR2. If this bit is '1', it is output to DDR Timing. If it is '0', it is output to DDR2 Timing.
io_strength	[9:8]	R/W	mDDR drive strength selector
io_drivesel	[10]	R/W	Drive select
io_odtsel	[11]	R/W	On-Die-Termination(ODT) select
io_odten	[12]	R/W	Active high On-Die_Termination(ODT) enable

io_odten	io_odtsel	termination
0x0	0x0	no termination
0x0	0x1	150 ohm
0x1	0x0	75 ohm
0x1	0x1	50 ohm

io_strength	drive_strength
2'b00	one-eighth Drive
2'b01	Quarter Drive
2'b10	Half Drive
2'b11	Full Drive

io_drivesel	drive
2'b0	60% Drive(SSTL I)
2'b1	Full Drive(SSTL II)

### 8.2.4 DDR\_TIMING\_0

Register	Bits	Type	Description
tras	[4:0]	R/W	ACTIVATE to PRECHARGE cycle : tRAS x Frequency / 1000
trfc	[14:8]	R/W	REFRESH to ACTIVATE or to REFRESH interval cycle : tRFC x Frequency / 1000
trc	[20:16]	R/W	ACTIVATE to ACTIVATE cycle : tRC x Frequency / 1000

### 8.2.5 DDR\_TIMING\_1

Register	Bits	Type	Description
trcd	[3:0]	R/W	ACTIVATE to READ or WRITE delay cycle : tRCD x Frequency / 1000
trp	[7:4]	R/W	PRECHARGE period cycle : tRP x Frequency / 1000
trrd	[11:8]	R/W	ACTIVATE to ACTIVATE delay different bank cycle : tRRD x Frequency / 1000
twr	[15:12]	R/W	Write Recovery time cycle : tWR x Frequency / 1000
twtr	[18:16]	R/W	Interval WRITE to READ delay cycle : tWTR x Frequency / 1000
tmrd	[21:20]	R/W	LOAD MODE cycle time
tdqss	[23:22]	R/W	WRITE command to first DQS latching transition cycle(only LPDDR)

### 8.2.6 DDR\_TIMING\_2

Register	Bits	Type	Description
trefr	[15:0]	R/W	Average periodic refresh : tREFI x Frequency / 1000
init_time	[31:16]	R/W	Stable power-up and clock cycle

### 8.2.7 DDR\_LMR\_EXT\_STD

Register	Bits	Type	Description
slmr	[14:0]	R/W	Mode Register
elmr	[30:16]	R/W	Extended Mode Register

### 8.2.8 DDR\_LMR\_EXT\_3\_2

Register	Bits	Type	Description
elmr2	[14:0]	R/W	Extended Mode Register 2 (only DDR2)
elmr3	[30:16]	R/W	Extended Mode Register 3 (only DDR2)

### 8.2.9 DDR\_CFG\_RDQ\_LATENCY

Register	Bits	Type	Description
Latency	[4:0]	R/W	It is Clock Cycle value at which Read Data is output after Command Address input. A Read Data Valid signal is generated by this register value. When this register is set, the value set in DDR_STS_RDQ_LATENCY can be known.
Align	[10:8]	R/W	It is a register for sorting Byte for Read Data. When this register is set, the value set in DDR_STS_RDQ_LATENCY can be known.

### 8.2.10 DDR\_CFG\_RDQ\_DLYVAL

Register	Bits	Type	Description
rdq_dlyval_0	[4:0]	R/W	Read DQ Byte Lane0 This register is used to manually set Delay Cell Value.
rdq_dlyval_1	[12:8]	R/W	Read DQ Byte Lane1 This register is used to manually set Delay Cell Value.
rdq_dlyval_2	[20:16]	R/W	Read DQ Byte Lane2 This register is used to manually set Delay Cell Value.
rdq_dlyval_3	[28:24]	R/W	Read DQ Byte Lane3 This register is used to manually set Delay Cell Value.

If you set it before Auto Cablibration, it will be applied immediately and you should not access it during the process. When the DDR Controller is activated after Auto Calibration is completed, the value set in this register is applied when Auto Refresh Timing is used. The completed state can be determined by the rdqdlyval\_pending bit of DDR\_STS\_RDQ\_LATENCY. The setting value of this register can be confirmed through DDR\_STS\_RDQ\_DLYVAL.  
[ Auto Calibration : Before > Immediate, After > auto refresh, Doing > no access ]

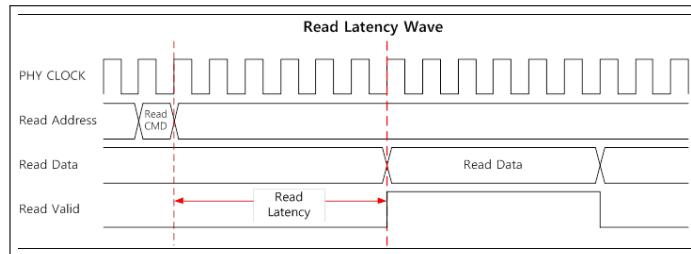
### 8.2.11 DDR\_CFG\_CMD\_DLYVAL

Register	Bits	Type	Description
clk_dlyval	[4:0]	R/W	Clock This register is used to manually set the cell value.
cmd_dlyval	[12:8]	R/W	Command and Address This register is used to manually set the cell value.
wdq_dlyval	[20:16]	R/W	Write DQ, DM, DQS It is a register that can set Delay Cell Value manually.

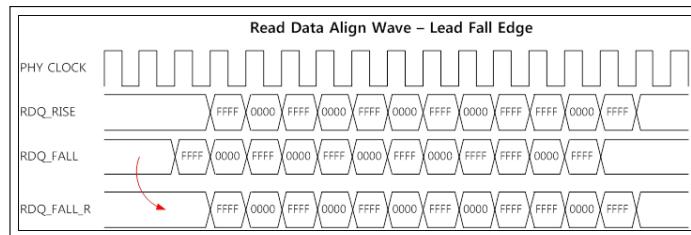
If you set it before Auto Cablibration, it will be applied immediately and you should not access it during the process. When DDR Control is activated after Auto Calibration is completed, the value set in this register is applied when Auto Refresh Timing. The completed state can be determined by the cmddlyval\_pending bit of DDR\_STS\_RDQ\_LATENCY. The setting value of this register can be confirmed through DDR\_STS\_CMD\_DLYVAL.  
[ Auto Calibration : Before > Immediate, After > auto refresh, Doing > no access ]

### 8.2.12 DDR\_STS\_RDQ\_LATENCY

Register	Bits	Type	Description
Latency	[4:0]	R	It is Clock Cycle value at which Read Data is output after Command Address input. A Read Data Valid signal is generated by this register value.
Align	[10:8]	R	It is a register indicating the sorted state of the data(rise/fall). 0 : capture data begin at rising edge 1 : capture date begin at falling edge
err_align	[17]	R	Align_Level_Error is a bit indicating this.
err_latency	[18]	R	Latency_Level_Error is a bit indicating this.
rdqdlyval_pending	[19]	R	When accessing the DDR_CFG_RDQ_DLYVAL register, this bit is set to '1' and immediately becomes '0' before Auto Calibration. After Auto Calibration is completed, it becomes '0' when Auto Refresh Timing is completed. If this bit is '0', the setting value of DDR_CFG_RDQ_DLYVAL is applied.
cmddlyval_pending	[20]	R	When accessing the DDR_CFG_RDQ_DLYVAL register, this bit is set to '1' and immediately becomes '0' before Auto Calibration. After Auto Calibration is completed, it becomes '0' when Auto Refresh Timing is completed. If this bit is '0', the setting value of DDR_CFG_CMD_DLYVAL is applied.



**Figure 8.1:** Read Latency Wave



**Figure 8.2:** Read Data Align Wave

### **8.2.13 DDR\_STS\_RDQ\_DLYVAL**

Register	Bits	Type	Description
sts_rdqddlyval_0	[4:0]	R	Indicates the delay cell value applied to byte lane 0.
sts_rqddlyval_1	[12:8]	R	Indicates the delay cell value applied to byte lane 1.
sts_rqddlyval_2	[20:16]	R	Indicates the delay cell value applied to byte lane 2.
sts_rqddlyval_3	[28:24]	R	Indicates the delay cell value applied to byte lane 3.

### **8.2.14 DDR\_STS\_CMD\_DLYVAL**

Register	Bits	Type	Description
sts_clkddlyval	[4:0]	R	Displays the delay cell value applied to the DDR clock.
sts_cmddlyval	[12:8]	R	Displays the status of delay cell applied to Command and Address
sts_wdqddlyval	[20:16]	R	Write Indicates the delay cell value status applied to DQ, DM, and DQS

### **8.2.15 DDR\_STS\_VREF\_ERRPOS**

Register	Bits	Type	Description
sts_verf_errpos	[9:0]	R	Auto Calibration Progress If error occurs during the verification process, the error position is displayed.
sts_verf_error	[16]	R	Auto Calibration Progress If the error occurs during the verification process, the error status is displayed. 0x0 : No Error 0x1 : Error

### **8.2.16 DDR\_STS\_VREF\_ERRDATA\_L**

Register	Bits	Type	Description
sts_verf_errdata	[31:0]	R	Auto Calibration Progress Error Data is displayed when an error occurs during the verification process.

### **8.2.17 DDR\_STS\_VREF\_ERRDATA\_H**

Register	Bits	Type	Description
sts_verf_errdata	[31:0]	R	Auto Calibration Progress Error Data is displayed when an error occurs during the verification process.

### 8.2.18 DDR\_STS\_RDQCPT\_R\_BEAT\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

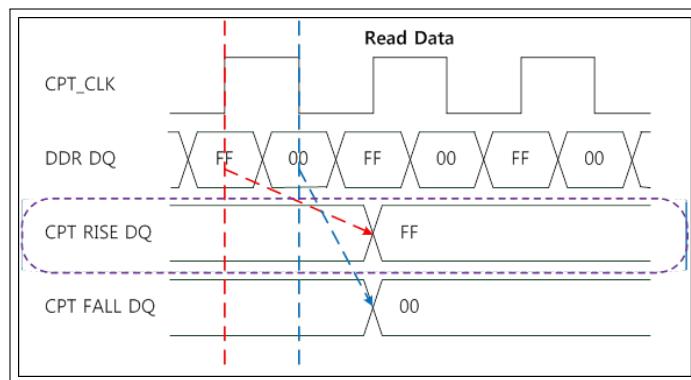


Figure 8.3: Read Data

### 8.2.19 DDR\_STS\_RDQCPT\_R\_BEAT\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.20 DDR\_STS\_RDQCPT\_R\_BEAT\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.21 DDR\_STS\_RDQCPT\_R\_BEAT\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.22 DDR\_STS\_RDQCPT\_F\_BEAT\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vale '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

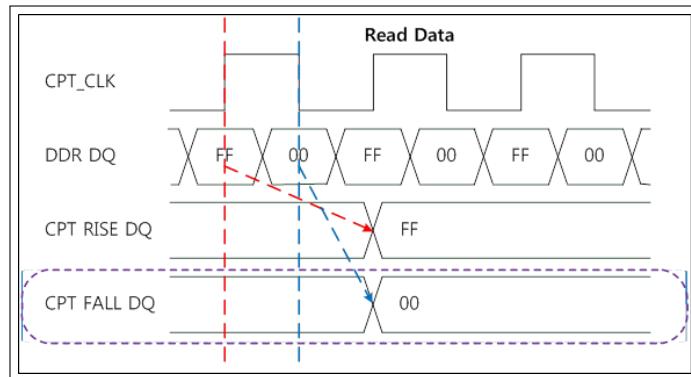


Figure 8.4: Read Data

### 8.2.23 DDR\_STS\_RDQCPT\_F\_BEAT\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.24 DDR\_STS\_RDQCPT\_F\_BEAT\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.25 DDR\_STS\_RDQCPT\_F\_BEAT\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.26 DDR\_STS\_RDQCPT\_BEAT\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

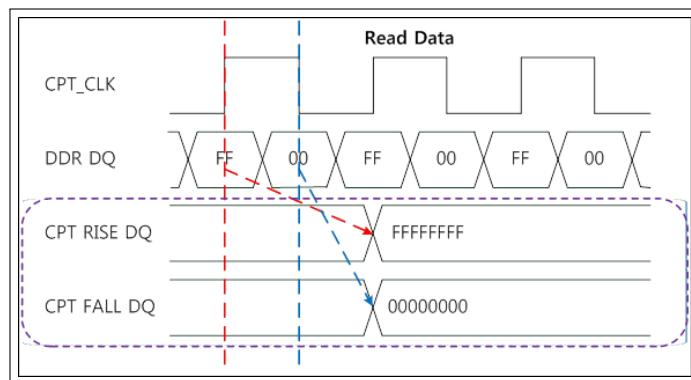


Figure 8.5: Read Data

### 8.2.27 DDR\_STS\_RDQCPT\_BEAT\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.28 DDR\_STS\_RDQCPT\_BEAT\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.29 DDR\_STS\_RDQCPT\_BEAT\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.30 DDR\_STS\_RDQCPT\_BYTE0\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

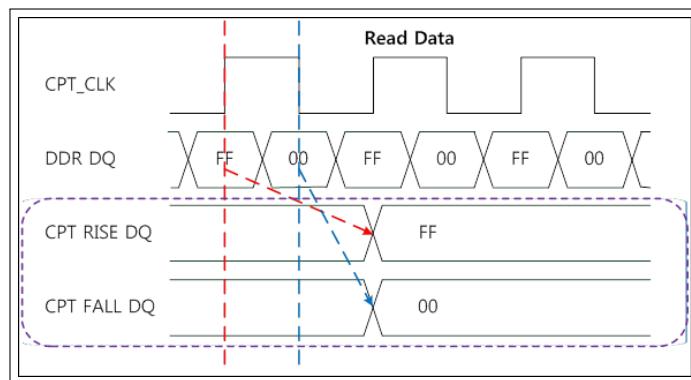


Figure 8.6: Read Data

### 8.2.31 DDR\_STS\_RDQCPT\_BYTE0\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.32 DDR\_STS\_RDQCPT\_BYTE0\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.33 DDR\_STS\_RDQCPT\_BYTE0\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.34 DDR\_STS\_RDQCPT\_BYTE1\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.35 DDR\_STS\_RDQCPT\_BYTE1\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.36 DDR\_STS\_RDQCPT\_BYTE1\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.37 DDR\_STS\_RDQCPT\_BYTE1\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.38 DDR\_STS\_RDQCPT\_BYTE2\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.39 DDR\_STS\_RDQCPT\_BYTE2\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.40 DDR\_STS\_RDQCPT\_BYTE2\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.41 DDR\_STS\_RDQCPT\_BYTE2\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.42 DDR\_STS\_RDQCPT\_BYTE3\_0\_7

Register	Bits	Type	Description
sts_rdqcpt_31	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '31'.
sts_rdqcpt_30	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '30'.
sts_rdqcpt_29	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '29'.
sts_rdqcpt_28	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '28'.
sts_rdqcpt_27	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '27'.
sts_rdqcpt_26	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '26'.
sts_rdqcpt_25	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '25'.
sts_rdqcpt_24	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '24'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.43 DDR\_STS\_RDQCPT\_BYTE3\_8\_15

Register	Bits	Type	Description
sts_rdqcpt_23	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '23'.
sts_rdqcpt_22	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '22'.
sts_rdqcpt_21	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '21'.
sts_rdqcpt_20	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '20'.
sts_rdqcpt_19	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '19'.
sts_rdqcpt_18	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '18'.
sts_rdqcpt_17	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '17'.
sts_rdqcpt_16	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '16'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.44 DDR\_STS\_RDQCPT\_BYTE3\_16\_23

Register	Bits	Type	Description
sts_rdqcpt_15	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '15'.
sts_rdqcpt_14	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '14'.
sts_rdqcpt_13	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '13'.
sts_rdqcpt_12	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '12'.
sts_rdqcpt_11	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '11'.
sts_rdqcpt_10	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '10'.
sts_rdqcpt_9	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '9'.
sts_rdqcpt_8	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '8'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

### 8.2.45 DDR\_STS\_RDQCPT\_BYTE3\_24\_31

Register	Bits	Type	Description
sts_rdqcpt_7	[1:0]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '7'.
sts_rdqcpt_6	[5:4]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '6'.
sts_rdqcpt_5	[9:8]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '5'.
sts_rdqcpt_4	[13:12]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '4'.
sts_rdqcpt_3	[17:16]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '3'.
sts_rdqcpt_2	[21:20]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '2'.
sts_rdqcpt_1	[25:24]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '1'.
sts_rdqcpt_0	[29:28]	R	This register is used to inform Read DQ Capture status of Delay Cell Vaule '0'.

Value	DQ Capture Value
0	{DW{1'b0}}
1	{DW{1'b1}}
2	Error
3	Unused

## 8.3 Activation GUIDE

- Register Setting Sequence
  - write DDR\_ADDR\_SIZE register
    - ex) nx\_cpuif\_reg\_write\_one( DDRC\_REG\_4 , 0x8C ); // address : bank, row, column
  - write DDR\_TIMING\_0 register
    - ex) nx\_cpuif\_reg\_write\_one( DDRC\_REG\_8 , 0x0B270 ); // for 200MHz operation
  - write DDR\_PHY\_CONFIG register
    - ex) nx\_cpuif\_reg\_write\_one( DDRC\_REG\_0 , 0x1 );
  - Wait DDR\_PHY\_CONFIG register initializing
    - ex) while(0 == nx\_cpuif\_reg\_read\_one(DDRC\_REG\_0, NULL) ); // wait 0x30

# 9 HOVER\_DMA

## 9.1 Features

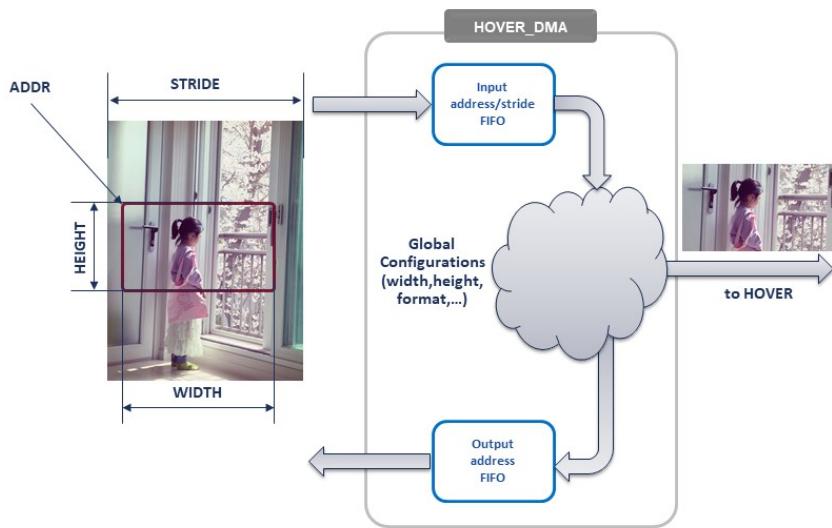
The HOVER DMA features include the following :

- Supports 60K x 60K pixels/frame
- Supports 60K byte stride
- Supports non-aligned base address
- Generating video signals for HOVER Format
- Input address/stride FIFO (4depth) If the input FIFO is empty, it will display using the last used address
- Output address FIFO (4depth) for software implementation convenience If an address is no longer used, the address is returned via this output FIFO
- Supports user programmable interrupt
- 32bit APB BUS for SFR
- 128 bit AXI4 BUS for 2D image reading 32bit Address

## 9.2 Functional Description

### 9.2.1 Frame buffer handling

HOVER DMA has an input address/stride FIFO for multiple buffer handling.



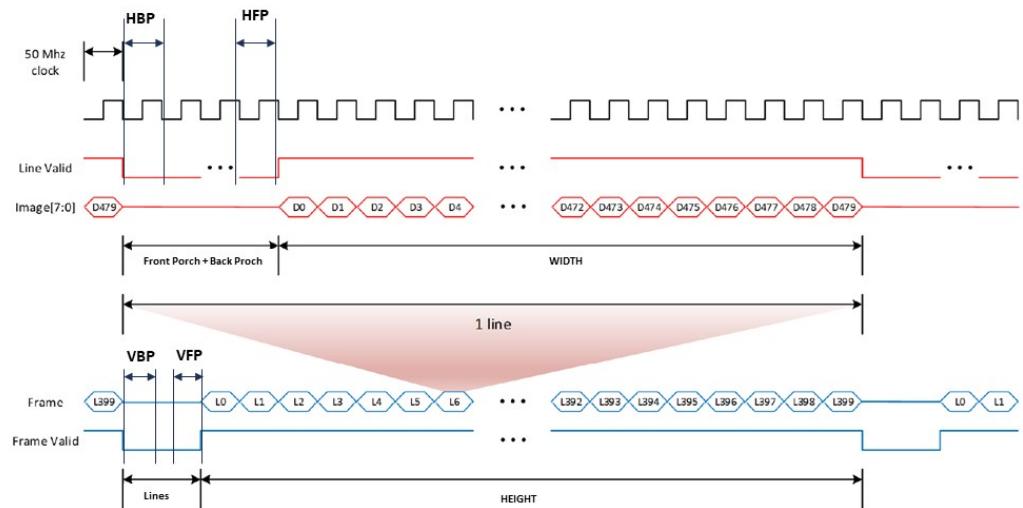
8 NEXELL

When a frame read completes for display, if there is an address/stride pair in the Input FIFO, the address/stride pair is used to read the next frame. In this case, if the address/stride pair is different from the last pair, DMA4DSIM pushes the last used address to the Output address FIFO.

If the Input FIFO is empty or the input address/stride pair is the same as the last pair, DMA4DSIM reads the next frame using the last pair. In this case, the input pair will be popped from the input FIFO.

Do not change display settings such as width, height, and format during operation. The behavior in such cases is undefined.

### 9.2.2 Waveform



## 9.3 HOVER DMA Register Description

### 9.3.1 Register Map Summary

- Base Address: 20D0\_0000(DMA4DSIM)

Register	Offset	Description	Reset Value
INT_ENABLE	0x0	Interrupt enable register Read: Interrupt enable status Write 1 : enable interrupt Write 0 : NOP	0x00000000
INT_DISABLE	0x4	Interrupt disable register Read: Interrupt enable status Write 1 : disable interrupt Write 0 : NOP	0x00000000
INT_PEND	0x8	Interrupt pending register Read: Interrupt pending status Write 1 : clear interrupt pending Write 0 : NOP	0x00000000
INT_STATUS	0xC	Interrupt state register Read: Interrupt pending status Write 1 : clear interrupt pending Write 0 : NOP	0x00000000
INT_PEND_NUMBER	0x10	Interrupt pend number register	0x00000000
CONTROL	0x14	Control register Read: Last written value Write 1 : do operation Write 0 : NOP	0x00000000
CONFIG	0x18	Configuration register	0x00000000
STATUS	0x1C	Status register	0x00000000
ADDR	0x20	Image address register	0x00000000
STRIDE	0x24	Image stride register (unit : byte) STRIDE = address of 2nd line - address of 1st line	0x00000000
WIDTH	0x28	Image width register (unit : pixel) If this value is changed while syncgen is running, the behavior is undefined It should be a multiple of 4	0x00000000
HEIGHT	0x2C	Image height register (unit : line) If this value is changed while syncgen is running, the behavior is undefined	0x00000000
HFP	0x30	Horizontal Front Porch register	0x00000000
HBP	0x34	Horizontal Back Porch register	0x00000000
VFP	0x38	Vertical Front Porch register	0x00000000
VBP	0x3C	Vertical Back Porch register	0x00000000
DONE_ADDR	0x40	Done buffer address	0x00000000

### 9.3.2 INT\_ENABLE

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x0 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
RSVD	[31:12]	-	RESERVED. IT SHOULD BE KEPT 0 DO NOT WRITE ANY 1	-
SYNCGEN_UNDERFLOW	[11]	R/W	[State] SYNCGEN is underflow	0x0
DONE_VALID	[10]	R/W	[State] DONE_VALID is HIGH	0x0
IN_READY	[9]	R/W	[State] IN_READY is HIGH	0x0
(IN_VALID&IN_READY)	[8]	R/W	[State] Both IN_VALID and IN_READY are HIGH	0x0
!SYNCGEN_IDLE	[7]	R/W	[State] Sync generator busy	0x0
SYNCGEN_IDLE	[6]	R/W	[State] Sync generator idle	0x0
negedge_SYNCGEN_IDLE	[5]	R/W	[Event] SYNCGEN IDLE negedge	0x0
posedge_SYNCGEN_IDLE	[4]	R/W	[Event] SYNCGEN IDLE posedge	0x0
!HOVER_DMA_CORE_IDLE	[3]	R/W	[State] HOVER DMA CORE is IDLE	0x0
HOVER_DMA_CORE_IDLE	[2]	R/W	[State] HOVER DMA CORE is not IDLE	0x0
negedge_HOVER_DMA_CORE_IDLE	[1]	R/W	[Event] negedge when HOVER DMA CORE is IDLE	0x0
posedge_HOVER_DMA_CORE_IDLE	[0]	R/W	[Event] posedge when HOVER DMA CORE is IDLE	0x0

### 9.3.3 INT\_DISABLE

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x4 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
INT_DISABLE	[31:0]	R/W	Interrupt disable register Read: Interrupt enable status Write 1 : disable interrupt Write 0 : NOP	0x0

### 9.3.4 INT\_PEND

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x8 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
INT_PEND	[31:0]	R/W	Interrupt pending register Read: Interrupt pending status Write 1 : clear interrupt pending Write 0 : NOP	0x0

### 9.3.5 INT\_STATUS

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0xC Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
INT_STATUS	[31:0]	R/W	Interrupt status register Read: Interrupt pending status Write 1 : clear interrupt pending Write 0 : NOP	0x0

### 9.3.6 INT\_PEND\_NUMBER

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x10 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
INT_PEND_NUMBER	[31:0]	R	Interrupt pend number register	0x0

### 9.3.7 CONTROL

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x14 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	RESERVED IT SHOULD BE KEPT 0, DO NOT WRITE ANY 1	-
IS_LEFT	[4]	R/W	0 : RIGHT ; 1 : LEFT	0x0
SYNCGEN_STOP	[3]	R/W	Sync gen stop	0x0
SYNCGEN_START	[2]	R/W	Sync gen run	0x0
DONE_READY	[1]	R/W	DONE_READY HIGH	0x0
IN_VALID	[0]	R/W	IN_VALID HIGH	0x0

### 9.3.8 CONFIG

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x18 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	RESERVED IT SHOULD BE KEPT 0, DO NOT WRITE ANY 1	-
auto clear DONE_FIFO	[1]	R/W	clear FIFO which is DONE	0x0
auto issue buffer	[0]	R/W	Buffer automatically issues	0x0

### 9.3.9 STATUS

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x1C Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	RESERVED IT SHOULD BE KEPT 0, DO NOT WRITE ANY 1	-
DONE_VALID	[4]	R/W	DONE_VALID is HIGH	0x0
IN_VALID	[3]	R/W	IN_VALID is HIGH	0x0
SYNCGEN_IDLE	[2]	R/W	SYNCGEN is IDLE	0x0
HOVER_DMA_CORE_IDLE	[1]	R/W	HOVER DMA CORE is IDLE	0x0
HOVER_DMA_CORE_IDLE & SYNCGEN_IDLE	[0]	R/W	HOVER DMA CORE and SYNCGEN both idle	0x0

### 9.3.10 ADDR

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x20 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
ADDR	[31:0]	R/W	Image address register	0x0

### 9.3.11 STRIDE

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x24 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
STRIDE	[31:0]	R/W	Image stride register (unit : byte) STRIDE = address of 2nd line - address of 1st line	0x0

### 9.3.12 WIDTH

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x28 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
WIDTH	[31:0]	R/W	Image width register (unit : pixel) If this value is changed while syncgen is running, the behavior is undefined It should be a multiple of 4	0x0

### 9.3.13 HEIGHT

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x2C Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
HEIGHT	[31:0]	R/W	Image height register (unit : line) If this value is changed while syncgen is running, the behavior is undefined	0x0

### 9.3.14 HFP

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x30 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
HFP	[31:0]	R/W	Horizontal Front Porch register	0x0

### 9.3.15 HBP

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x34 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
HBP	[31:0]	R/W	Horizontal Back Porch register	0x0

### 9.3.16 VFP

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x38 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
VFP	[31:0]	R/W	Vertical Front Porch register	0x0

### 9.3.17 VBP

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x3C Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
VBP	[31:0]	R/W	Vertical Back Porch register	0x0

### 9.3.18 DONE\_ADDR

- Base Address: 20D0\_0000(DMA4DSIM)
- Address = Base Address + 0x40 Reset Value = 0x00000000

Name	Bit	Type	Description	Reset Value
DONE_ADDR	[31:0]	R/W	Done buffer address	0x0

**10 VIP**

## 10.1 Overview

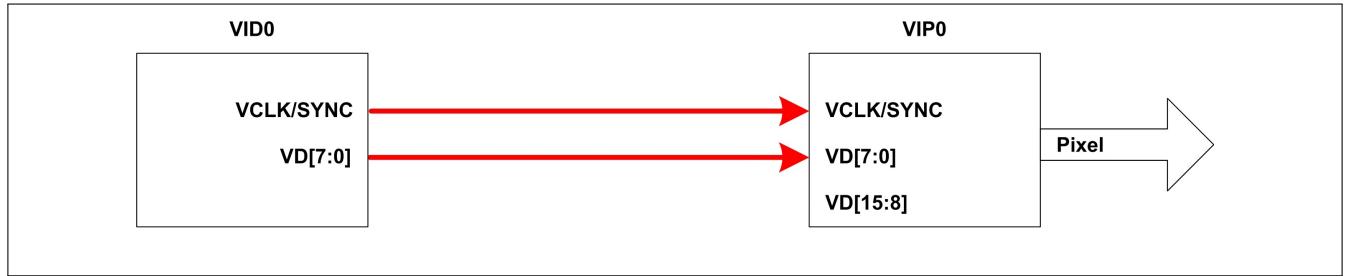
The Video Input Processor (hereinafter VIP) of the DRONE\_SOC can receive images directly from external camera modules or video decoders. In addition, it can clip or scale down the input images and store them to the memory. The images stored from the VIP can be used for preview images by using the Multi-Layer Controller (MLC).

## 10.2 Features

- Supports ITU-BT601, ITU-BT656(YUV 422 & RAW).
- Clock, HSync, VSync, Field, 8bit data ports.
- Maximum 8192x8192 image Supports.
- Supports 3-Plane YUV 4:2:0, 4:2:2, 4:4:4, LinearYUYV, 8Bit RAW
- Clipping and Scale Down.
- Vertical and Operation Done Interrupt.
- 1 Inputs from External CIS.

## 10.3 Interconnection

### 10.3.1 External Pad Interconnection

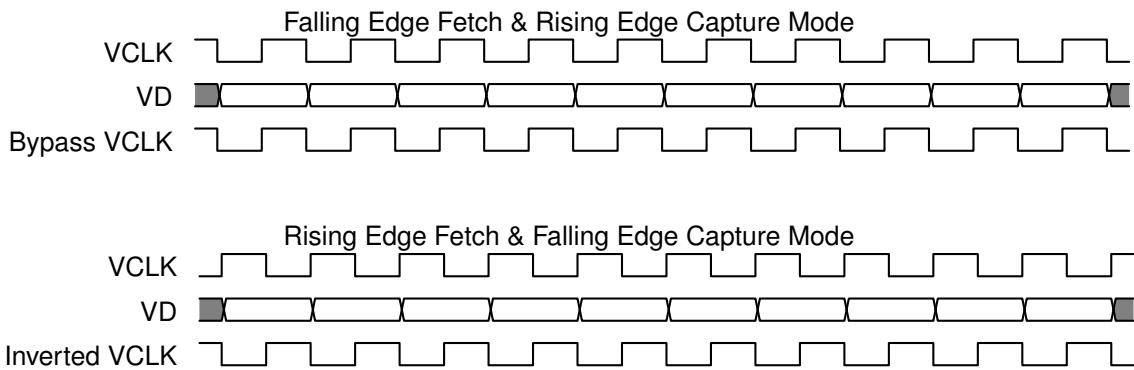


**Figure 10.1:** VIP External Interface Interconnect

The VIP Interconnected with 8-bit External CIS modules as shown as in the above figure 10.1.

### 10.3.2 Clock Generation

The VIP can create the video in clock by using an internal PLL or an external VCLK pin as a clock source. The created video in clock is used for sync signal creation and data interface in the Video Input Port block as shown in figure 10.2.



**Figure 10.2:** Clock Polarity settings

In general, the VIP is designed to capture data on the rising edge. Therefore, if the video clock fetches data at Rising edge, the VIP can invert input clock. Or if External CSI fetches video data at falling edge, should be bypass the input clock via VIP0\_PADCLKSEL.

## 10.4 Video Input Port

### 10.4.1 Block Diagram

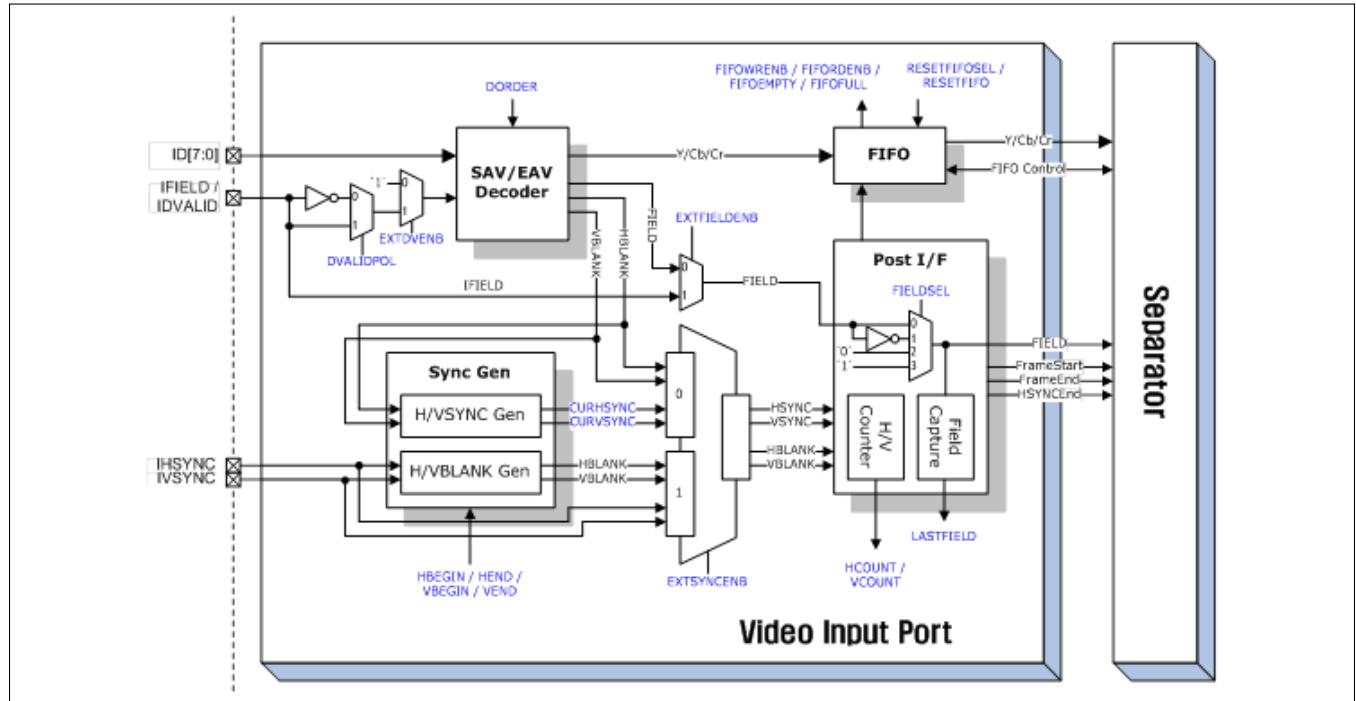


Figure 10.3: Video Input Port Block Diagram

### 10.4.2 Sync Generation

The horizontal and vertical timing interfaces for the video input port are as shown in the following Figure 10.4.

Each symbol in the above figure is described in the following table 10.1.

symbol	Brief	Remark
tHSW	Horizontal Sync Width	Number of VCLKs in the section wherer the horizontal sync pulse is active
tHPB	Horizontal Back Porch	Number of VCLKs in section from the end point of the horizontal sync pulse to the start point of the horizontal active
tHFP	Horizontal Front Porch	Number of VCLKs in a section from the end point of the horizontal sync pulse to the start point of the horizontal active
tAVW	Active Video Width	Number of VCLKs in a horizontal active section
tVSW	Vertical Sync Width	Number of lines in a section where the vertical sync pulse is active
tVBP	Vertical Back Porch	Number of lines in a section from the end point of the vertical sync pulse to the start point the next vertical active
tVFP	Vertical Front Porch	Number of lines in a section from the end point of the vertical active to the start point of the vertical sync pulse
tAVH	Active Video Height	Number of lines in the vertical active section

Table 10.1: Horizontal & Vertical Timing Symbols

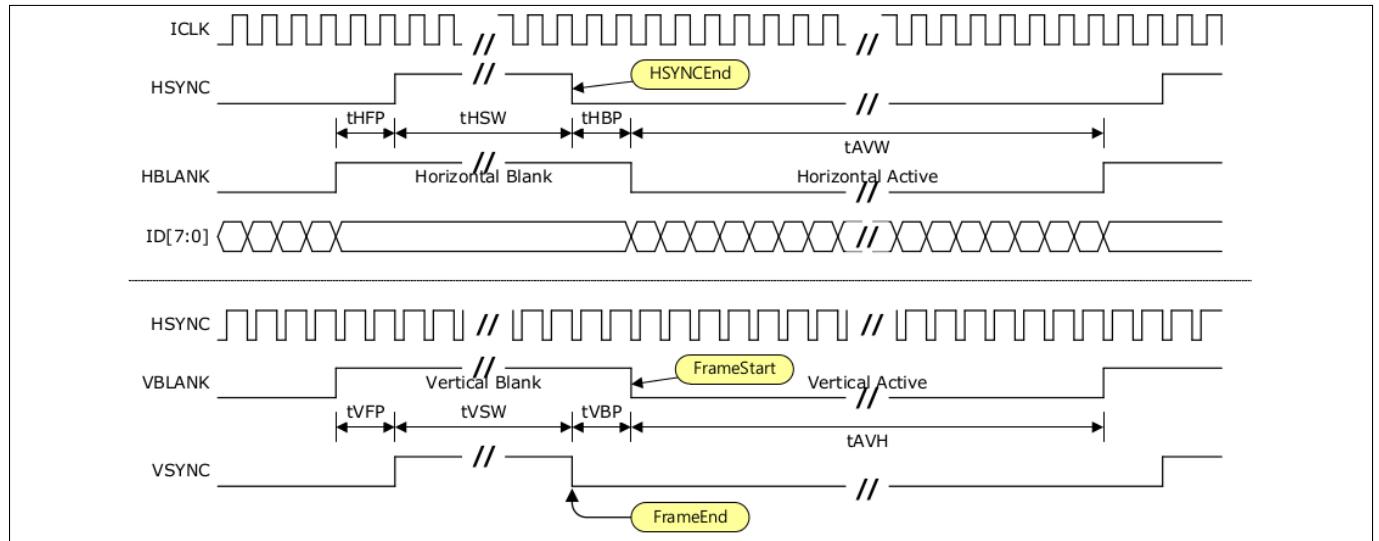


Figure 10.4: Horizontal &amp; Vertical Timing

#### 10.4.3 ITU-R BT.601

If the port uses an external HSYNC or VSYNC, the EXTSYNCENB bit should be set as 1. If the EXTSYNCENB bit is 1, the port receives the HSYNC and VSYNC from the outside, and creates the HBLANK and VBLANK internally. The polarity of the external H(V)SYNC only supports high active. In the following figure 10.5 shows the relationship between the HBLANK and the VBLANK generated from external HSYNC and VSYNC inputs.

The HBEGIN and the HEND are used to generate the HBLANK from the HSYNC. The VBEGIN and the VEND are used to generate the VBLANK from the VSYNC. The settings for each register are listed in the following Table 10.2.

Register	Formula	Remark
VBEGIN	$tVBP - 1$	Number of lines in a section from the end point of the VSYNC to the start point of the vertical active video - 1
VEND	$tVBP + tAVH - 1$	Number of lines in a section from the end point of the VSYNC to the end point of the vertical active video - 1
HBEGIN	$tHBP - 1$	Number of clocks in a section from the end point of the HSYNC to the start point of the horizontal active video - 1
HEND	$tHBP + tAVW - 1$	Number of clocks in a section from the end point of the HSYNC to the start point of the horizontal active video - 1

Table 10.2: Register Setting for ITU-R BT.601

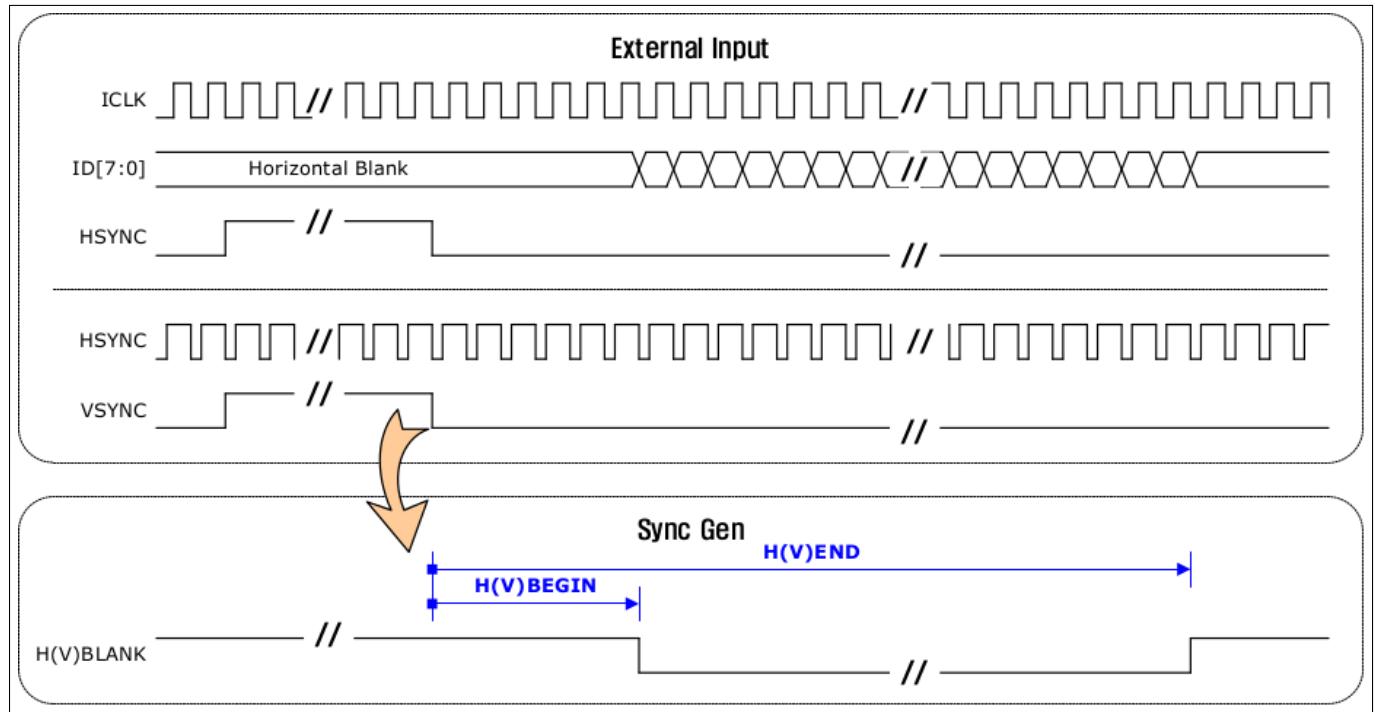


Figure 10.5: Sync Generation for ITU-R BT.601

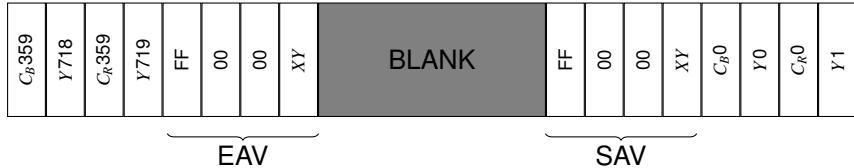


Figure 10.6: ITU-R BT.656 Multiplexed 8-Bit Data Stream Format with EAV/SAV

#### 10.4.4 ITU-R BT.656

In the ITU-R BT.656 format, there is no additional sync signal pin, and the sync information is transmitted along with data via data pins. At this time, the Sync information is inserted as an additional code before the start point of the valid data (SAV) and after the end of the valid data (EAV). Sync information included in data is as shown in the following Figure 10.6.

The SAV and the EAV consists of [FF, 00, 00, XY]. Each code contains Field (F), VSYNC (V) and HSYNC (H) data, and each code is composed as follows table 10.3.

To create the HBLANK and VBLANK from the sync information contained in the data, the EXTSYNCENB should be set as 0. The SAV/EAV decoder blocks generate the HBLANK and VBLANK from the sync information contained in the data. The Sync Gen block generates the HSYNC and the VSYNC based on the HBLANK and VBLANK signals. In the following figure 10.7 shows the relationship that generates the H(V)BLANK and H(V)SYNC from the SAV/EAV contained in the data. If an external CSI using BT.1120 format, the VIP\_MXS\_ENB should be set as 1.

If the EXTSYNCENB is 0, the HBEGIN and HEND\* are used to generate the HSYNC signal from the HBLANK. The VBEGIN and VEND are used to generate the VSYNC signal from the VBLANK. The settings for each register are

Function	1(MSB)	F	V	H	P3	P2	P1	P0(LSB)	HEX	Brief Description
0	1	0	0	0	0	0	0	0	80h	SAV of odd field
1	1	0	0	1	1	1	0	1	9Dh	EAV of odd field
2	1	0	1	0	1	0	1	1	ABh	SAV of odd blank
3	1	0	1	1	0	1	1	0	B6h	EAV of odd blank
4	1	1	0	0	0	1	1	1	C7h	SAV of even Field
5	1	1	0	1	1	0	1	0	DAh	EAV of even Field
6	1	1	1	0	1	1	0	0	ECh	SAV of even blank
7	1	1	1	1	0	0	0	1	F1h	EAV of even blank

F: Field select ( 0: odd(first) field, 1: even(second) field)V: Vertical Blanking (0: Active, 1: Blank)H: SAV/EAV (0:SAV, 1:EAV)

Table 10.3: Embedded Sync Code

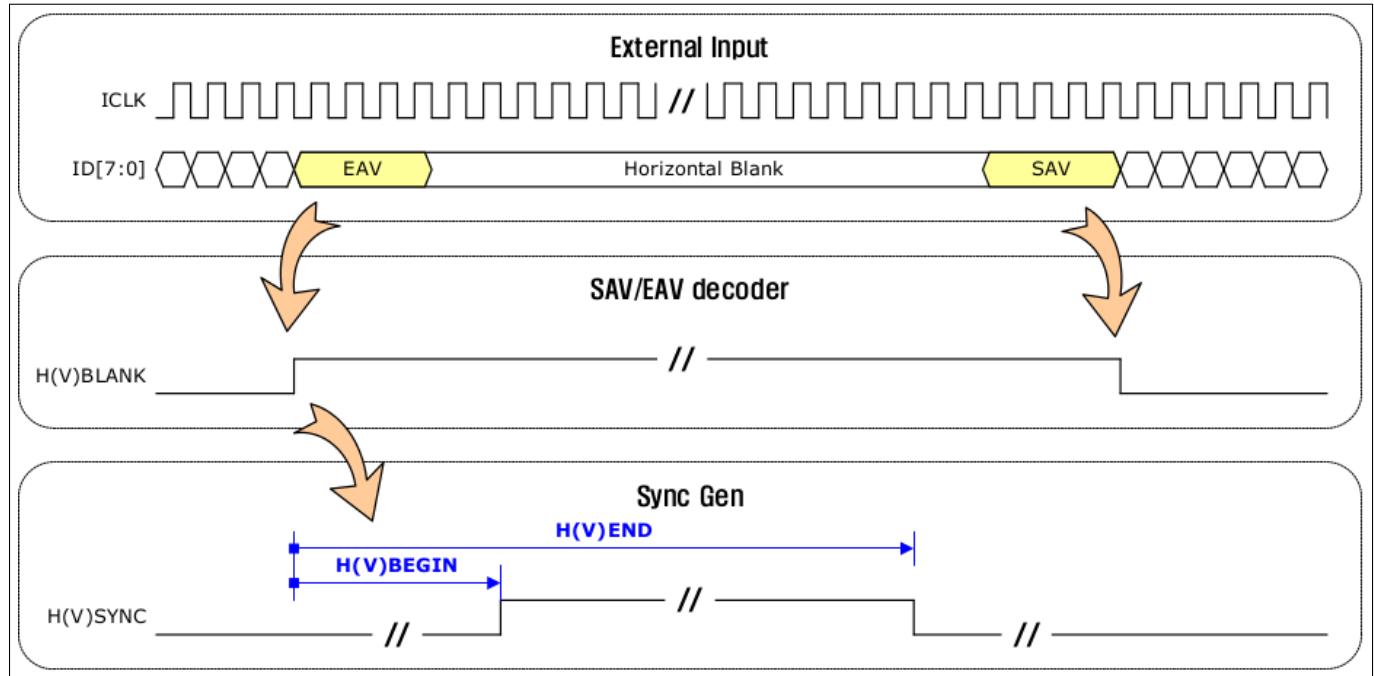


Figure 10.7: Sync Generation for ITU-R BT.656

listed in the following table 10.4.

Register	Formula	Remark
VBEGIN	tVFP + 1	Number of lines in a section from the end point of the vertical active video to the start point of the VSYNC - 1
VEND	tVFP + tVSW + 1	Number of lines in a section from the end point of the vertical active video to the end point of the VSYNC + 1
HBEGIN	tHFP - 7	Number of clocks in a section from the end point of the horizontal active video to the start point of the HSYNC - 7
HEND	tHFP + tHSW - 7	Number of clocks in a section from the end point of the horizontal active video to the end point of the HSYNC - 7

**Table 10.4:** Register Settings for ITU-R BT.656

## 10.4.5 External Data Valid and Field

The video input port can receive data valid signals or field signals from the outside. Since the IDVALID and the IFIELD share a pin, users can use only one of them.

### 10.4.5.1 External Data Valid

If the EXTDVENB is set as 1, users can use the input signal from the IFIELD/IDVALID pad as the IDVALID signal. In this case, the polarity is determined by the DVALIDPOL. The video input port is designed to use the IDVALID signal of active high mode internally. Therefore, if the polarity of an external IDVALID signal is active low, the input signal should be inverted by setting the DVALIDPOL as 0. If the polarity of an external IDVALID signal is active high, the input signal should be bypassed by setting the DVALIDPOL as 1. Even though an external IDVALID signal is used, the internal HBLANK and VBLANK signals are used. Therefore, the user should set the H(V)BEGIN and the H(V)END.

### 10.4.5.2 External Field

If an external field signal is used, the EXTFIELDENB should be set as 1. In the ITU-R BT.656 format, the input signal from the IFIELD/IDVALID pad can be used as an external field signal by setting the EXTFIELDENB as 1. The video input port internally considers it as an odd field if the polarity of the field signal is low. If the polarity of a field signal is high, the port considers it as an even field. The user can select the polarity of a field signal by using FIELDSEL, or can fix the polarity as 0 or 1.

### 10.4.5.3 Data Order

The video input port can select the order of input data. Basically, the ITU-R BT.656 format or the ITU-R BT.601 8-bit format has the order [Cb, Y0, Cr, Y1]. If the order of the input data is different from the default order, users can change the order via DORDER. The data orders supported by the video input port are listed in the following table 10.5.

DORDER	1st	2nd	3rd	4th
0	CB	Y0	CR	Y1
1	CR	Y0	CB	Y1
2	Y0	CB	Y1	CR
3	Y0	CR	Y1	CB

**Table 10.5:** VIP Data Order

#### 10.4.5.4 Horizontal & Vertical Counter

The video input port can inform the user of the size of the active section. The HCOUNT indicates the total clock numbers of a line in the active video section. The VCOUNT indicates the total line numbers in the active video section.

#### 10.4.5.5 Current HSYNC & VSYNC Status

When the EXTSYNCENB is 0 the CURHSYNC and the CURVSYNC bits are provided to show the HSYNC and VSYNC states.

#### 10.4.5.6 Current Field Status

LASTFIELD is updated whenever a field signal is changed. In addition, when a VSYNC interrupt occurs, the user can get the field status of the next data by using LASTFIELD.

#### 10.4.5.7 FIFO Controls

The video input port block can inform the user of the current status of the internal FIFO. The FIFOWRENB indicates if data is being written to the FIFO. The FIFORDENB indicates if data is being read from the FIFO. If the FIFO is empty, the FIFOEMPTY is set as 1. If the FIFO is full, the FIFOFULL is set as 1. In addition, users can reset the FIFO at a specific point. According to the RESETFIFOSEL setting, the reset point of the FIFO can be controlled by selecting either FrameEnd (the end of VSYNC, RESETFIFOSEL is 0), FrameStart (the start of vertical active video, RESETFIFOSEL is 1) or the RESETFIFO bit (RESETFIFOSEL is 2), or by selecting all of them (RESETFIFOSEL is 3). The RESETFIFO bit is valid only when the RESETFIFOSEL is 2 or 3. If the FIFO is reset by setting the RESETFIFO as 1, the RESETFIFO should be set as 0 again.

#### 10.4.5.8 Recommend Setting for Video Input Port

In the following table 10.6 lists the recommend settings for the video input port by input formats.

Register	ITU-R BT.656	ITU-R BT.601 P	ITU-R BT601 I
EXTSYNCENB	0	1	1
DWIDTH	1	1	1
DORDER	0	0	0
EXTFIELDENB	0	0	1
FIELDSEL	0	3	0 or 1
EXTDVENB	0	0 or 1	0 or 1
DVALIDPOL	Not Used	0 or 1	0 or 1
VBEGIN	tVFP + 1	tVBP - 1	tVBP - 1
VEND	tVFP + tVSW + 1	tVBP + tAVH - 1	tVBP + tAVH - 1
HBEGIN	tHFP - 7	tHBP - 1	tHBP - 1
HEND	tHFP + tHSW - 7	tHBP + tAVW = 1	tHBP + tAVW = 1

**Table 10.6:** Recommend Setting for Video Input Port

## 10.5 Clipper & Decimator

### 10.5.1 Clipping & Scale Down

The VIP can store input images to the memory after clipping or scaling down. An input image is transmitted to the Clipper through the video input port and the Separator. The Clipper clips a specific area from the input image, and then stores the result in the memory and transmits the result to the Decimator. The Decimator can store the image transmitted from the Clipper in the memory after scaling down the image. 10.8 shows the procedure for clipping and scaling down an input image.

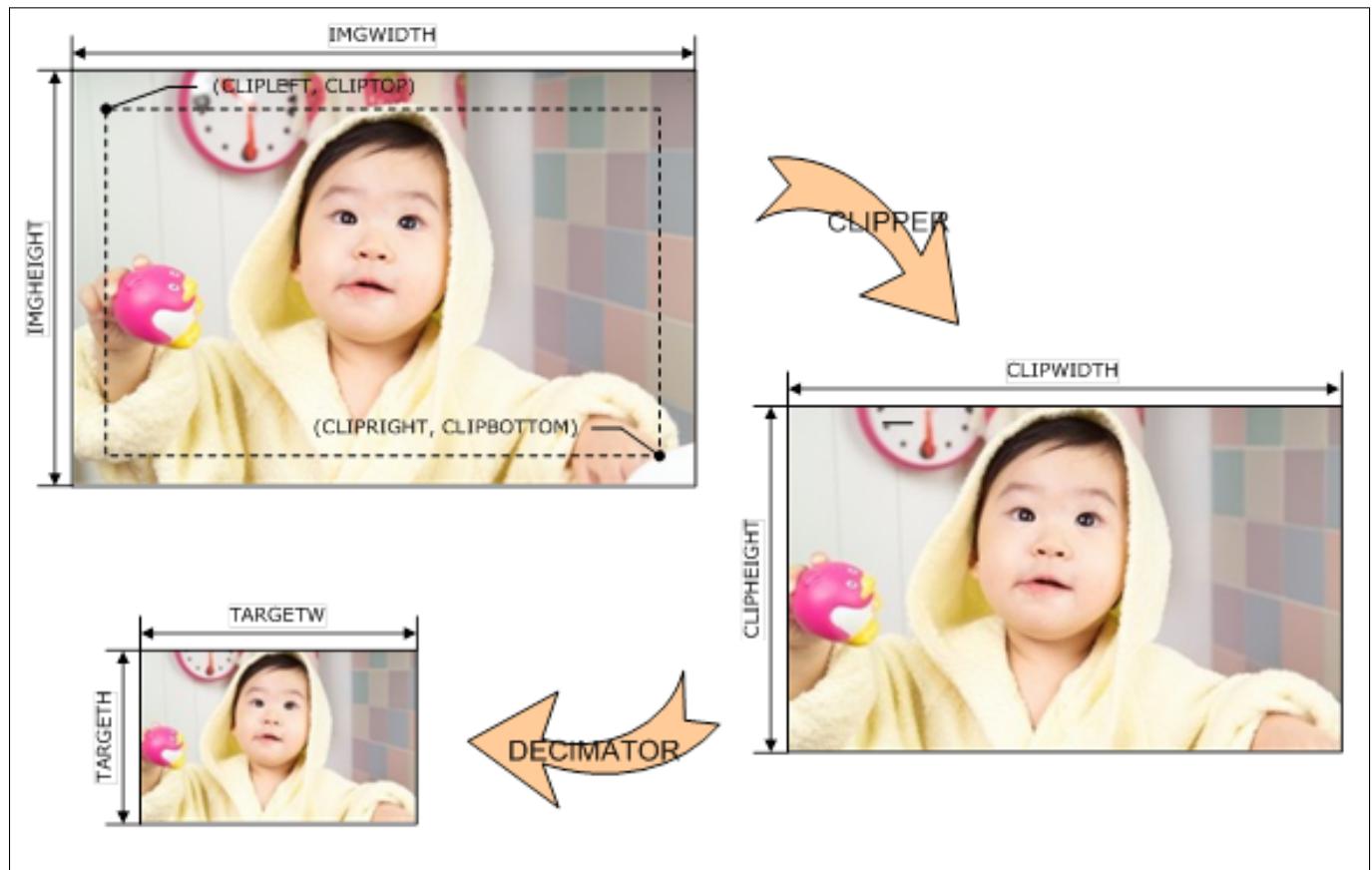


Figure 10.8: Clipping & Decimation

Users can enable the Clipper to clip an input image by specifying the relevant area, using CLIPLEFT, CLIPRIGHT, CLIPTOP and CLIPBOTTOM. The Decimator scales down the clipped image by using the Bresenham algorithm. To this end, TARGETW, TARGETH, DELTAW, DELTAH, CLEARW, and \*CLEARH\* are used. In the following table lists the settings for each register.

Register	Formula	Range	Unit	Remark
TARGETW	-	0 to 8191	Pixel	Width of a scaled-down image
TARGETH	-	0 to 8191	Pixel	Height of scaled-down image
DELTAW	CLIPWIDTH - TARGETW	0 to 8191	Pixel	Width difference between the original image and the result image
DELAH	CLIPHEIGHT-TARGETH	0 to 8191	Pixel	Height difference between the original image and the result image
CLEARW	TARGETW-DELTAW	0 to 8191	Pixel	Difference between the width of the result image and the DELTAW
CLEARH	TARGETH-DELAH	0 to 8191	Pixel	Difference between the height of the result image and the DELTAH

**Table 10.7:** Registers for Scaling

### 10.5.2 Interlace Scan mode

If the INTERLACENB is set as '1' for interlace scan mode, the Clipper and Decimator store output images by field signals. The Clipper and Decimator can also select the polarity of a field signal by using FIELDINV. The Clipper and Decimator supports frame-based output images, and outputs the images after automatically adjusting the start address and the start line depending on the field signal.

### 10.5.3 Output Data Format

The VIP can store input images in separated YUV format 4:2:0, 4:2:2, 4:4:4 format. Users can select either 4:2:0, 4:2:2 or 4:4:4 format by using FORMATSEL.

### 10.5.4 Output Address and Stride

The VIP has 6 BASEADDR and STRIDE. It is used to calculate the addresses of Y, U, V of Clipper and Decimator respectively. The STRIDE has the byte size of a line. When X, Y are the horizontal and vertical coordinates of the video, each transaction address is output as a result of BASEADDR+(Y\*STRIDE)+X. Since this Y value is automatically doubled in interlace scan mode, the value in progressive scan mode should be specified.

### 10.5.5 Interrupt Generation

The VIP has three interrupt sources. The three interrupt sources are the HSINT, which generates at the end of a horizontal sync, the VSINT, which generates an interrupt at the end of a vertical sync, and the ODINT, which generates an interrupt when the Clipper and Decimator operations are finished. The Pending bits and the Enable bits for each interrupt exist, and each register is listed in the following table 10.8.

Interrupt	Enable bit	Pending bit	Condition
HSINT	HSINTENB	HSINTPEND	End of horizontal sync pulse
VSINT	VSINTENB	VSINTPEND	End of vertical sync pulse
ODINT	ODINTENB	ODINTPEND	End of frame and bus transaction

**Table 10.8:** Interrupt registers

## 10.6 VIP Register Map

### 10.6.1 Register Map Summary

- Base Address: 20400000

Register	Offset	Description	Reset Value
VIP0_CONFIG	0x00	VIP0 Configuration Register	0x0000_0000
VIP0_INTCTRL	0x04	VIP0 Interrupt Control Register	0x0000_0000
VIP0_FIELD	0x08	VIP0 Field Control Register	0x0000_0000
VIP0_SYNCMON	0x0C	VIP0 Sync Monitor Register	0x0000_0000
VIP0_VBEGIN	0x10	VIP0 Vertical Sync Start Register	0x0000_0000
VIP0_VEND	0x14	VIP0 Vertical Sync End Register	0x0000_0000
VIP0_HBEGIN	0x18	VIP0 Horizontal Sync Start Register	0x0000_0000
VIP0_HEND	0x1C	VIP0 Horizontal Sync End Register	0x0000_0000
VIP0_FIFOCTRL	0x20	VIP0 FIFO Control Register	0x0000_0000
VIP0_HCOUNT	0x24	VIP0 Horizontal Counter Register	0x0000_0000
VIP0_VCOUNT	0x28	VIP0 Vertical Counter Register	0x0000_0000
VIP0_PADCLK_INV	0x30	VIP0 PAD Clock Inverter Register	0x0000_0000
VIP0_INFIFO_CLR	0x34	VIP0 IN-FIFO Clear Register	0x0000_0000

### 10.6.2 VIP0\_CONFIG

- Base Address: 20400000
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
EXTSYNCENB	[8]	R/W	External Sync Enable. 0: Not use External Sync (656 Format) 1: Using External Sync(601 format)	0
RSVD	[7:4]	-	Reserved	-
DORDER	[3:2]	R/W	Specifies the order of input video data. 00 : Cb, Y0, Cr, Y1 (or RAW Data) 01 : Cr, Y1, Cb, Y0 10 : Y0, Cb, Y1, Cr 11 : Y1, Cr, Y0, Cb( or Separated RAW Data )	0
DWIDTH	[1]	R/W	Specifies the bit-width of an input video signal 0: 16 bit 1: 8 bit	0
VIPENB	[0]	R/W	VIP0 Enable 0 : Disable 1 : Enable	0

### 10.6.3 VIP0\_INTCTRL

- Base Address: 20400000
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:10]	-	Reserved	-
HSYNC_INTENB	[9]	R/W	H-sync Interrupt enable register	0
VSYNC_INTENB	[8]	R/W	V-sync Interrupt enable register	0
RSVD	[7:2]	-	Reserved	-
HSYNC_INTPEND	[1]	R/W	H-sync Interrupt pending register	0
VSYNC_INTPEND	[0]	R/W	V-sync Interrupt pending register	0

### 10.6.4 VIP0\_FIELD

- Base Address: 20400000
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:13]	-	Reserved	-
VIP0_VSYNCGEN-SOURCE	[12]	R/W	Vsync Gen Source.	0
VIP0_EXTVBLANKMODE	[11]	R/W	External Vblank Mode	0
VIP0_EXTH-BLANKMODE	[10]	R/W	External Hblank Mode	0
VIP0_EXTVSYNCPOL	[9]	R/W	External Vsync Polarity	0
VIP0_EXTHSYNCPOL	[8]	R/W	External Hsync Polarity	0
RSVD	[7:6]	-	Reserved	-

VIP0_LASTFIELD	[5]	R	Indicates the status of the internal Field signal that is updated at every FrameStart (the start of vertical active video). For the operation of this bit, the PCLKMODE is set as 1. 0 : The last field is an odd field. 1 : The last field is an even field.	0
VIP0_DVALIDPOL	[4]	R/W	Selects the polarity of an external DVALID signal. This bit is valid only when the EXTDVENB is 1. 0 : Active Low 1 : Active High	0
VIP0_EXTFIELDENB	[3]	R/W	Specifies the use of an external field signal. 0 : Disable 1 : Enable	0
VIP0_EXTDVENB	[2]	R/W	Specifies the use of an external DVALID signal. 0 : Disable 1 : Enable	0
VIP0_FIELDSEL	[1:0]	R/W	Selects a field signal. 00 : Bypass (Low is odd field) 01 : Invert (Low is even field) 10 : Fix 0 (odd field) 11 : Fix 1 (even field)	0

### 10.6.5 VIP0\_SYNCMON

- Base Address: 20400000
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
VIP0_CURHSYNC	[1]	R	Indicates the status of the VSYNC created by the internal sync generator in Embedded Sync mode. This bit is valid only when the EXTSYNCENB is 0. 0 : Inactivate 1 : Activate	0
VIP0_CURVSYNC	[0]	R	Indicates the status of the VSYNC created by the internal sync generator in Embedded Sync mode. This bit is valid only when the EXTSYNCENB is 0. 0 : Inactivate 1 : Activate	0

### 10.6.6 VIP0\_VBEGIN

- Base Address: 20400000
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_VBEGIN	[15:0]	R/W	When the EXTSYNCENB is 1, this value is used for the creation of an internal vertical blank. This value specifies the number of lines in a section from the end of the vertical sync pulse to the end of the vertical blank. VBEGIN = tVBP ? 1 When the EXTSYNCENB is 0, this value is used for the creation of an internal vertical sync pulse. This value specifies the number of lines in a section from the start of the vertical blank to the start of the vertical sync pulse. VBEGIN = tVFP + 1	0

### 10.6.7 VIP0\_VEND

- Base Address: 20400000
- Address = Base Address + 0x14 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_VEND	[15:0]	R/W	<p>When the EXTSYNCENB is 1, this value is used for the creation of an internal vertical blank. This value specifies the number of lines in a section from the end of the vertical sync pulse to the start of the vertical blank.</p> <p>VBEGIN = tVBP + tAVH ? 1</p> <p>When the EXTSYNCENB is 0, this value is used for the creation of an internal vertical sync pulse. This value specifies the number of lines in a section from the start of the vertical blank to the end of the vertical sync pulse.</p> <p>VBEGIN = tVFP + tVSW + 1</p>	0

### 10.6.8 VIP0\_HBEGIN

- Base Address: 20400000
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_HBEGIN	[15:0]	R/W	<p>When the EXTSYNCENB is 1, this value is used for the creation of an internal horizontal blank. This value specifies the number of clocks in a section from the end of the horizontal sync pulse to the end of the horizontal blank.</p> <p>HBEGIN = tHBP ? 1</p> <p>When the EXTSYNCENB is 0, this value is used for the creation of an internal horizontal sync pulse. This value specifies the number of clocks in a section from the start of the horizontal blank to the start of the horizontal sync pulse.</p> <p>HBEGIN = tHFP ? 7</p>	0

### 10.6.9 VIP0\_HEND

- Base Address: 20400000
- Address = Base Address + 0x1C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_HEND	[15:0]	R/W	<p>When the EXTSYNCENB is 1, this value is used for the creation of an internal horizontal blank. This value specifies the number of clocks in a section from the end of the horizontal sync pulse to the start of the horizontal blank.</p> <p>HBEGIN = tHBP + tAVW ? 1</p> <p>When the EXTSYNCENB is 0, this value is used for the creation of an internal horizontal sync pulse. This value specifies the number of clocks in a section from the start of the horizontal blank to the end of the horizontal sync pulse.</p> <p>HBEGIN = tHFP + tHSW - 7</p>	0

### 10.6.10 VIP0\_FIFOCTRL

- Base Address: 20400000
- Address = Base Address + 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:12]	-	Reserved	-
VIP0_FIFOWRENB	[11]	R	Indicates the data write status of the internal FIFO of the VIP. 0 : No write 1 : Writing	0
VIP0_FIFORDENB	[10]	R	Indicates the data read status of the internal FIFO of the VIP. 0 : No read 1 : Reading	0
VIP0_FIFOEMPTY	[9]	R	Indicates whether the internal FIFO of the VIP0 is empty or not. 0 : Not empty 1 : Empty	0
VIP0_FIFOFULL	[8]	R	Indicates whether the internal FIFO of the VIP0 is full or not. 0 : Not full 1 : Full	0
RSVD	[7:3]	-	Reserved	-
VIP0_RESETFIFOSEL	[2:1]	R/W	Controls the point at which the FIFO of the VIP0 is reset. 00 : FrameEnd (the end of the vertical sync pulse) 01 : FrameStart (the start of the vertical active video) 10 : RESETFIFO bit (Clear by user) 11 : ALL (FrameEnd or FrameStart or RESETFIFO bit)	0
VIP0_RESETFIFO	[0]	R/W	Resets the internal FIFO of the VIP. This bit should be reset as 0 after being set as 1, and it is valid only when the RESETFIFOSEL is 2 or 3. 0 : Release FIFO Reset 1 : Reset FIFO	0

### 10.6.11 VIP0\_HCOUNT

- Base Address: 20400000
- Address = Base Address + 0x24 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_HCOUNT	[15:0]	R	Indicates the total number of clocks in the horizontal active video section. When EXTSYNCENB is 0, this value has horizontal active video clocks + 4.	0

### 10.6.12 VIP0\_VCOUNT

- Base Address: 20400000
- Address = Base Address + 0x28 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_VCOUNT	[15:0]	R	Indicates the total number of lines in the vertical active video section.	0

### 10.6.13 VIP0 Pad clock invert

- Base Address: 20400000
- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
VIP0_PADCLKSEL	[2]	R/W	In-clock Invert. 0: Bypass the clock 1: Invert the clock	0
RSVD	[1:0]	-	Reserved	-

### 10.6.14 VIP0 INFIFO Clear

- Base Address: 20400000
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
GLITCH_REMOVE	[2]	R/W	should be set 1.	0
RSVD	[1]	-	Reserved	-
INFIFO_CLR	[0]	R/W	Input Async FIFO Clear	0

## 10.7 VIP Prescaler Register Map

### 10.7.1 Register Map Summary

- Base Address: 20400000

Register	Offset	Description	Reset Value
VIP_CDENB	0x200	VIP0 Clipper & Decimator Enable Register	0x0000_0000
VIP_ODINT	0x204	VIP0 Operation Done Interrupt Register	0x0000_0000
VIP_IMGWIDTH	0x208	VIP0 Image Width Register	0x0000_0000
VIP_IMGHEIGHT	0x20C	VIP0 Image Height Register	0x0000_0000
CLIP_LEFT	0x210	VIP0 Clipper Left Register	0x0000_0000
CLIP_RIGHT	0x214	VIP0 Clipper Right Register	0x0000_0000
CLIP_TOP	0x218	VIP0 Clipper Top Register	0x0000_0000
CLIP_BOTTOM	0x21C	VIP0 Clipper Bottom Register	0x0000_0000
DECI_TARGETW	0x220	VIP0 Decimator Target Width Register	0x0000_0000
DECI_TARGETH	0x224	VIP0 Decimator Target Height Register	0x0000_0000
DECI_DELTAW	0x228	VIP0 Decimator Delta Width Register	0x0000_0000
DECI_DELTAH	0x22C	VIP0 Decimator Delta Height Register	0x0000_0000
DECI_CLEARW	0x230	VIP0 Decimator Clear Width Register	0x0000_0000
DECI_CLEARH	0x234	VIP0 Decimator Clear Height Register	0x0000_0000
DECI_FORMAT	0x244	VIP0 Decimator Format Register	0x0000_0000
DECI_LUADDR	0x248	VIP0 Decimator LU Address Register	0x0000_0000
DECI_LUSTRIDE	0x24C	VIP0 Decimator LU Stride Register	0x0000_0000
DECI_CRADDR	0x250	VIP0 Decimator CR Address Register	0x0000_0000
DECI_CRStride	0x254	VIP0 Decimator CR Stride Register	0x0000_0000
DECI_CBADDR	0x258	VIP0 Decimator CB Address Register	0x0000_0000
DECI_CBStride	0x25C	VIP0 Decimator CB Stride Register	0x0000_0000
CLIP_FORMAT	0x288	VIP0 Clipper Format Register	0x0000_0000
CLIP_LUADDR	0x28C	VIP0 Clipper LU Address Register	0x0000_0000
CLIP_LUSTRIDE	0x290	VIP0 Clipper LU Stride Register	0x0000_0000
CLIP_CRADDR	0x294	VIP0 Clipper CR Address Register	0x0000_0000
CLIP_CRStride	0x298	VIP0 Clipper CR Stride Register	0x0000_0000
CLIP_CBADDR	0x29C	VIP0 Clipper CB Address Register	0x0000_0000
CLIP_CBStride	0x2A0	VIP0 Clipper CB Stride Register	0x0000_0000
VIP_SCANMODE	0x2C0	VIP0 Scan Mode Register	0x0000_0000
VIP_PORTSEL	0x2D8	VIP0 Scan Mode Register	0x0000_0000

### 10.7.2 VIP\_CDENB

- Base Address: 20400000
- Address = Base Address + 0x200 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
VIP0_REG_CLEAR	[319]	W	VIP CPUIF Register Clear	0
VIP0_SEPENB	[8]	R/W	Enables/disables the Separator. 0 : Disable 1 : Enable	0
RSVD	[7:2]	-	Reserved	-
VIP0_CLIPENB	[1]	R/W	Enables/disables the memory writing function of the Clipper block. This bit is valid only when the SEPENB is 1. 0 : Disable 1 : Enable	0
VIP0_DECIENB	[0]	R/W	Enables/disables the memory writing function of the Decimator block. This bit is valid only when the SEPENB is 1. 0 : Disable 1 : Enable	0

### 10.7.3 VIP\_ODINT

- Base Address: 20400000
- Address = Base Address + 0x204 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
OD_INTENB	[8]	R/W	Operation Done Interrupt Enable	0
RSVD	[7:1]	-	Reserved	-
OD_INTPEND	[0]	R/W	Operation Done Interrupt Pend	0

### 10.7.4 VIP\_IMGWIDTH

- Base Address: 20400000
- Address = Base Address + 0x208 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_IMGWIDTH	[14:0]	R/W	Specifies the width of input images in pixel units. When EXSYN-CENB is 0, you have to set it as image width + 2.	0

### 10.7.5 VIP\_IMGHEIGHT

- Base Address: 20400000
- Address = Base Address + 0x20C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_IMGHEIGHT	[14:0]	R/W	Specifies the height of input images in line units.	0

### 10.7.6 CLIP\_LEFT

- Base Address: 20400000
- Address = Base Address + 0x210 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_CLIPLEFT	[14:0]	R/W	Specifies the X-coordinate on the top left corner of the area to be clipped, in pixels. The clipping width (CLIPRIGHT - CLIPLEFT) must be a multiple of 16.	0

### 10.7.7 CLIP\_RIGHT

- Base Address: 20400000
- Address = Base Address + 0x214 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_CLIPRIGHT	[14:0]	R/W	Specifies the X-coordinate on the bottom right corner of the area to be clipped, in pixels. It should have a greater value than the CLIPLEFT. The clipping width (CLIPRIGHT - CLIPLEFT) must be a multiple of 16.	0

### 10.7.8 CLIP\_TOP

- Base Address: 20400000
- Address = Base Address + 0x218 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_CLIPTOP	[14:0]	R/W	Specifies the Y-coordinate on the top left corner of the area to be clipped, in pixels. The clipping height (CLIPBOTTOM - CLIPTOP) must be an even number.	0

### 10.7.9 CLIP\_BOTTOM

- Base Address: 20400000
- Address = Base Address + 0x21C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_CLIPBOTTOM	[14:0]	R/W	Specifies the Y-coordinate on the bottom right corner of the area to be clipped, in pixels. It should have a greater value than the CLIPTOP. The clipping height (CLIPBOTTOM - CLIPTOP) must be an even number.	0

### 10.7.10 DECI\_TARGETW

- Base Address: 20400000
- Address = Base Address + 0x220 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_TARGETW	[14:0]	R/W	Specifies the width of the Decimator output image, in pixels. The width of the output image should be narrower than that of the clipped input image and be a multiple of 16.	0

### 10.7.11 DECI\_TARGETH

- Base Address: 20400000
- Address = Base Address + 0x224 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_TARGETH	[14:0]	R/W	Specifies the height of the Decimator output image, in lines. The height of the output image should be lower than that of the clipped input image and be an even number.	0

### 10.7.12 DECI\_DELTAW

- Base Address: 20400000
- Address = Base Address + 0x228 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_DELTAW	[14:0]	R/W	Specifies the width difference between the input image and the output image of the Decimator in pixel units. DELTAW = (CLIPRIGHT - CLIPLEFT) - TARGETW	0

### 10.7.13 DECI\_DELTAH

- Base Address: 20400000
- Address = Base Address + 0x22C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:15]	-	Reserved	-
VIP0_DELTAH	[14:0]	R/W	Specifies the line difference between the input image and the output image of the Decimator in line units. DETAH = (CLIPBOTTOM - CLIPTOP) - TARGETH	0

### 10.7.14 DECI\_CLEARW

- Base Address: 20400000
- Address = Base Address + 0x230 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_CLEARW	[15:0]	R/W	Specifies the difference between the width of the Decimator output image and the DELTAW in pixel units.This value has 2s complement format. CLEARW = TARGETW - DELTAW	0

### 10.7.15 DECI\_CLEARH

- Base Address: 20400000
- Address = Base Address + 0x234 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
VIP0_CLEARH	[15:0]	R/W	Specifies the difference between the height of the Decimator output image and the DELTAH in line units.This value has 2s complement format. CLEARH = TARGETH - DELTAH	0

### 10.7.16 DECI\_FORMAT

- Base Address: 20400000
- Address = Base Address + 0x244 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
DECI_FORMATSEL	[3:0]	R/W	Specifies the output format of the Decimator. 0000 = Separated YUV 4:2:0 0001 = Separated YUV 4:2:2 0010 = Separated YUV 4:4:4	0

### **10.7.17 DECI\_LUADDR**

- Base Address: 20400000
- Address = Base Address + 0x248 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DECI_LUADDR	[31:0]	R/W	Decimator LU Address	0

### **10.7.18 DECI\_LUSTRIDE**

- Base Address: 20400000
- Address = Base Address + 0x24C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DECI_LUSTRIDE	[31:0]	R/W	Decimator LU Stride	0

### **10.7.19 DECI\_CRADDR**

- Base Address: 20400000
- Address = Base Address + 0x250 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DECI_CRADDR	[31:0]	R/W	Decimator CR Address	0

### **10.7.20 DECI\_CRSTRIDE**

- Base Address: 20400000
- Address = Base Address + 0x254 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DECI_CRSTRIDE	[31:0]	R/W	Decimator CR Stride	0

### 10.7.21 DECI\_CBADDR

- Base Address: 20400000
- Address = Base Address + 0x258 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DECI_CBADDR	[31:0]	R/W	Decimator CR Address	0

### 10.7.22 DECI\_CBSTRIDE

- Base Address: 20400000
- Address = Base Address + 0x25C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DECI_CBSTRIDE	[31:0]	R/W	Decimator CB Stride	0

### 10.7.23 CLIP\_FORMAT

- Base Address: 20400000
- Address = Base Address + 0x288 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
CLIP_FORMATSEL	[3:0]	R/W	Specifies the output format of the Clipper. 0000 = Separated YUV 4:2:0 0001 = Separated YUV 4:2:2 0010 = Separated YUV 4:4:4	0

### 10.7.24 CLIP\_LUADDR

- Base Address: 20400000
- Address = Base Address + 0x28C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CLIP_LUADDR	[31:0]	R/W	Clipper LU Address	0

### 10.7.25 CLIP\_LUSTRIDE

- Base Address: 20400000
- Address = Base Address + 0x290 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CLIP_LUSTRIDE	[31:0]	R/W	Clipper LU Stride	0

### 10.7.26 CLIP\_CRADDR

- Base Address: 20400000
- Address = Base Address + 0x294 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CLIP_CRADDR	[31:0]	R/W	Clipper CR Address	0

### 10.7.27 CLIP\_CRStride

- Base Address: 20400000
- Address = Base Address + 0x298 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CLIP_CRStride	[31:0]	R/W	Clipper CR Stride	0

### 10.7.28 CLIP\_CBADDR

- Base Address: 20400000
- Address = Base Address + 0x29C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CLIP_CBADDR	[31:0]	R/W	Clipper CB Address	0

### 10.7.29 CLIP\_CBSTRIDE

- Base Address: 20400000
- Address = Base Address + 0x2A0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CLIP_CBSTRIDE	[31:0]	R/W	Clipper CB Stride	0

### 10.7.30 VIP\_SCANMODE

- Base Address: 20400000
- Address = Base Address + 0x2C0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
VIP0_INTERLACEENB	[1]	R/W	Specifies the scan mode of an input image. 0 : Progressive scan mode 1 : Interlace scan mode	0
VIP0_FIELDINV	[0]	R/W	Specifies the polarity of the field signal transmitted from the VIP0 block to the Clipper and to the Decimator. 0 : Bypass (Low is odd field) 1 : Invert (Low is even field)	0

### 10.7.31 VIP\_PORTSEL

- Base Address: 20400000
- Address = Base Address + 0x2D8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
VIP0_PORT_SELECTOR	[0]	R/W	VIP Port Selector 0: use VD 0 port 1: use VD 1 port (not implemented)	0

# 11 Multi-Format Video Codec

## 11.1 Overview

The multi format video codec (hereinafter referred to as "VPU") is a full HD multi-standard video IP for consumer multimedia products such as HDTVs, HD set-top boxes, and HD DVD players. It can decode compressed video in a format of H.264 BP/MP/HP, VC-1 SP/MP/AP, MPEG-1/2, MPEG-4 SP/ASP, H.263P3, VP8, Theora, AVS, RV-8/9/10, and JPEG (max. 8192 x 8192) It can also perform H.264, MPEG-4, and H.263 encoding up to Full-HD 1920 x 1088 (max. 8192 x 8192 JPEG) resolution. The VPU can perform simultaneous multiple real time encoding, decoding, or both encoding and decoding of different format video streams at multiple resolutions.

The VPU contains a 16-bit DSP called BIT processor. The BIT processor communicates with a host CPU through a host interface and controls the other sub-blocks of the VPU. The host CPU require slow resources under 1 MIPS, because all of the functions such as bit stream parsing, video hardware sub-blocks control and error resilience are implemented in the BIT processor. Moreover it is designed to optimally share most of the sub-blocks that are used in common for video processing, which contributes to the ultra low power and low gate count.

It is connected with a host CPU system via 32-bit AMBA 3 APB bus for system control and 128-bit AMBA3 AXI for data. There are two 128-bit AXI buses: primary and secondary. The secondary bus can be connected to on-chip memories to achieve high performance.

## 11.2 Functional Description

### 11.2.1 List of Video CODECs

The following table shows many different video standards supported by VPU.

-	<b>Standard</b>	<b>Profile</b>	<b>Level</b>	<b>Max.Rslt</b>	<b>Min.Rslt</b>	<b>Bitrate</b>
Encoder	H.264	Baseline	4	1920x1088	96x16	20Mbps
	MPEG-4	SP	5/6	1920x1088	96x16	20Mbps
	H.263	Profile3	70	1920x1088	96x16	20Mbps
Decoder	H.264	BP/MP/HP	4.2	1920x1088	16x16	50Mbps
	MPEG-4	ASP		1920x1088	16x16	40Mbps
	H.263	Profile3		1920x1088	16x16	20Mbps
	VC-1	SP/MP/AP	3	1920x1088	16x16	45Mbps
	MPEG-1/2	MP	High	1920x1088	16x16	80Mbps
	VP8			1920x1088	16x16	20Mbps
	Theora			1280x720	16x16	20Mbps
	AVS	Jizhun	6.2	1920x1088	16x16	40Mbps
	RV	8/9/10		1920x1088	16x16	40Mbps
Encoder	MJPEG	Baseline		8192x8192	16x16	160Mpel/sat YUV422
Decoder	MJPEG	Baseline		1920x1088	16x16	120Mpel/sat YUV444

**Table 11.1:** Supported Video Standards

## 11.2.2 Supported Video Encoding Tools

### 11.2.2.1 H.264/AVC BP/CBP Encoder

- Compatible with the ITU-T Recommendation H.264 specification
- The encoder uses only one reference frame for the motion estimation.
- 1/4-pel accuracy motion estimation with programmable search range up to [+128, +64]
- Search range is reconfigurable by SW
  - Horizontal (-128 to 127), Vertical (-64 to 63)
  - Horizontal (-64 to 63), Vertical (-32 to 31)
  - Horizontal (-32 to 31), Vertical (-16 to 15)
  - Horizontal (-16 to 15), Vertical (-16 to 15)
- 16 x 16, 16 x 8, 8 x 16 and 8 x 8 block sizes are supported.
- Available block sizes can be configurable.
- Intra-prediction
  - Luma I4 x 4 Mode: 9 modes
  - Luma I16 x 16 Mode: 3 modes (Vertical, Horizon, DC)
  - Chroma Mode: 3 modes (Vertical, Horizon, DC)
- Minimum encoding image size is 96 pixels in horizontal and 16 pixels in vertical.
- The encoder supports the following error resilience tools: video packet (fixed number of bits, and fixed number of macro blocks), CIR (Cyclic Intra Refresh), and multi-slice structure.
- FMO/ASO tool of H.264 is not supported.
- The encoder rate control is configurable for low-delay and long-delay, and configurable from macro block-level rate control to frame-level rate control.
- Field encoding is available without PAFF, MBAFF.

### 11.2.2.2 MPEG4-SP Encoder

- Compatible with the ISO/IEC 14496-2 specification
- MV with unrestricted motion vector
- AC/DC prediction
- 1/2-pel accuracy motion estimation with search range up to [+128, +64]
- Search range is reconfigurable by SW
  - Horizontal (-128 to 127), Vertical (-64 to 63)
  - Horizontal (-64 to 63), Vertical (-32 to 31)
  - Horizontal (-32 to 31), Vertical (-16 to 15)
  - Horizontal (-16 to 15), Vertical (-16 to 15)
- Error resilience tools such as re-sync marker, data-partitioning with reversible VLC.

### 11.2.2.3 H.263 P0/P3 (Interactive and Streaming Wireless Profile) Encoder

- MV with unrestricted motion vector mode compliant to Annex D
- Search range is -16 to 15 in horizontal and -16 to 15 in vertical
- H.263 Baseline profile + Annex J, K (RS = 0 and ASO = 0), and T

## 11.2.3 Supported Video Decoding Tools

### 11.2.3.1 H.264/AVC Decoder

- Fully compatible with the ITU-T Recommendation H.264 specification in BP, MP and HP
- Supports MVC Stereo High profile
- Supports CABAC/CAVLC
- Variable block size (16 x 16, 16 x 8, 8 x 16, 8 x 8, 8 x 4, 4 x 8 and 4 x 4)
- Error detection, concealment and error resilience tools with FMO/ASO support

### 11.2.3.2 VC-1/WMV-9 Decoder

- Supports all VC-1 profile features - SMPTE Proposed SMPTE Standard for Television: VC-1 Compressed
- Video Bit stream format and Decoding Process
- Supports Simple/Main/Advanced Profile
- Supports multi-resolution (Dynamic resolution) without scaling that returns related information

### 11.2.3.3 MPEG-4 Decoder

- Fully compatible with the ISO/IEC 14496-2 specification in SP/ASP except GMC(Global motion compensation)
- Full XviD compatibility
- Support for short video header

### 11.2.3.4 Sorenson Spark Decoder

- Fully compatible with Sorenson Spark decoder specification

### 11.2.3.5 H.263 V2 (Interactive and Streaming Wireless Profile, Profile 3) Decoder

- H.263 Baseline profile + Annex I, J, K (except RS/ASO), and T

### 11.2.3.6 MPEG-1/MPEG-2

- Fully compatible with ISO/IEC 13818-2 MPEG2 specification in Main Profile
- Support I, P and B frame
- Support field coded picture (interlaced) and fame coded picture

### 11.2.3.7 AVS Decoder

- Supports Jizhun profile level 6.2 (exclude 422 case)

### 11.2.3.8 Real Video 10 Decoder

- Fully compatible with RV-8/9/10 except re-sampling feature
- Minimum decoding size is 32 x 32 pixels.

### 11.2.3.9 VP8 Decoder

- Fully compatible with VP8 decoder specification
- Supporting both simple and normal in-loop de-blocking

### 11.2.3.10 Theora Decoder

- Fully compatible with Theora decoder specification

## 11.2.4 Supported JPEG Tools

### 11.2.4.1 MJPEG Baseline Process Encoder and Decoder

- Baseline ISO/IEC 10918-1 JPEG compliance
- Support 1 or 3 color components
- 3 component in a scan (interleaved only)
- 8-bit samples for each component
- Support 4:2:0, 4:2:2, 2:2:4, 4:4:4 and 4:0:0 color format (max. six 8 x 8 blocks in one MCU)
- Minimum encoding size is 16 x 16 pixels.

## 11.2.5 Non-codec related features

### 11.2.5.1 Value Added Features

- De-ringing (MPEG-2/4 only), rotator/mirroring
- Built-in de-blocking filter for MPEG-2/MPEG-4
- Pre/Post rotator/mirror

### 11.2.5.2 Programmability

- The VPU embeds 16-bit DSP processor dedicated to processing bit stream and controlling their video hardware.
- General purpose registers and interrupt for communication between a host processor and the video IP

### 11.2.5.3 Optimal External Memory Accesses

- Configurable frame buffer formats (linear or tiled) for longer burst-length
- 2D cache for motion estimation and compensation to reduce external memory accesses
- Secondary AXI port for on-chip memory to enhance performance

# 12 Scaler

## 12.1 Overview

The Scaler is the block to change image sizes. The Scaler reads an image from the memory and writes the image to the memory after Up/Down Scaling and Low-pass Filtering. At this time, the Scaler changes the direction of the image by using the Flip or Rotation function.

## 12.2 Features

### 12.2.1 Scaler

- Source/Destination Image
- Format: Separated YUV Format (420, 422, 444), Interleaved UV
- Size: (8 to 4096) x (8 to 4096) (Width is set as a multiple of eight).
- Upscale Ratio:  $8 \times 8 \rightarrow 4096 \times 4096$
- Downscale Ratio:  $4096 \times 4096 \rightarrow 8 \times 8$
- Lowpass filter available after Upscale or before Downscale.
- Horizontal 5-Tab Filter: Coefficients 64 Sets.
- Vertical 3-Tab Filter: Coefficients 32 Sets (For Frequency Response, refer to Operation Item).

## 12.3 Block Diagram

The Scaler consists of the blocks (SRC\_ADDR\_GEN, DEST\_ADDR\_GEN) to generate addresses, the Filter block, the FIFO block, and the blocks (CPUIF and POS\_GEN2) to exchange data with bus.

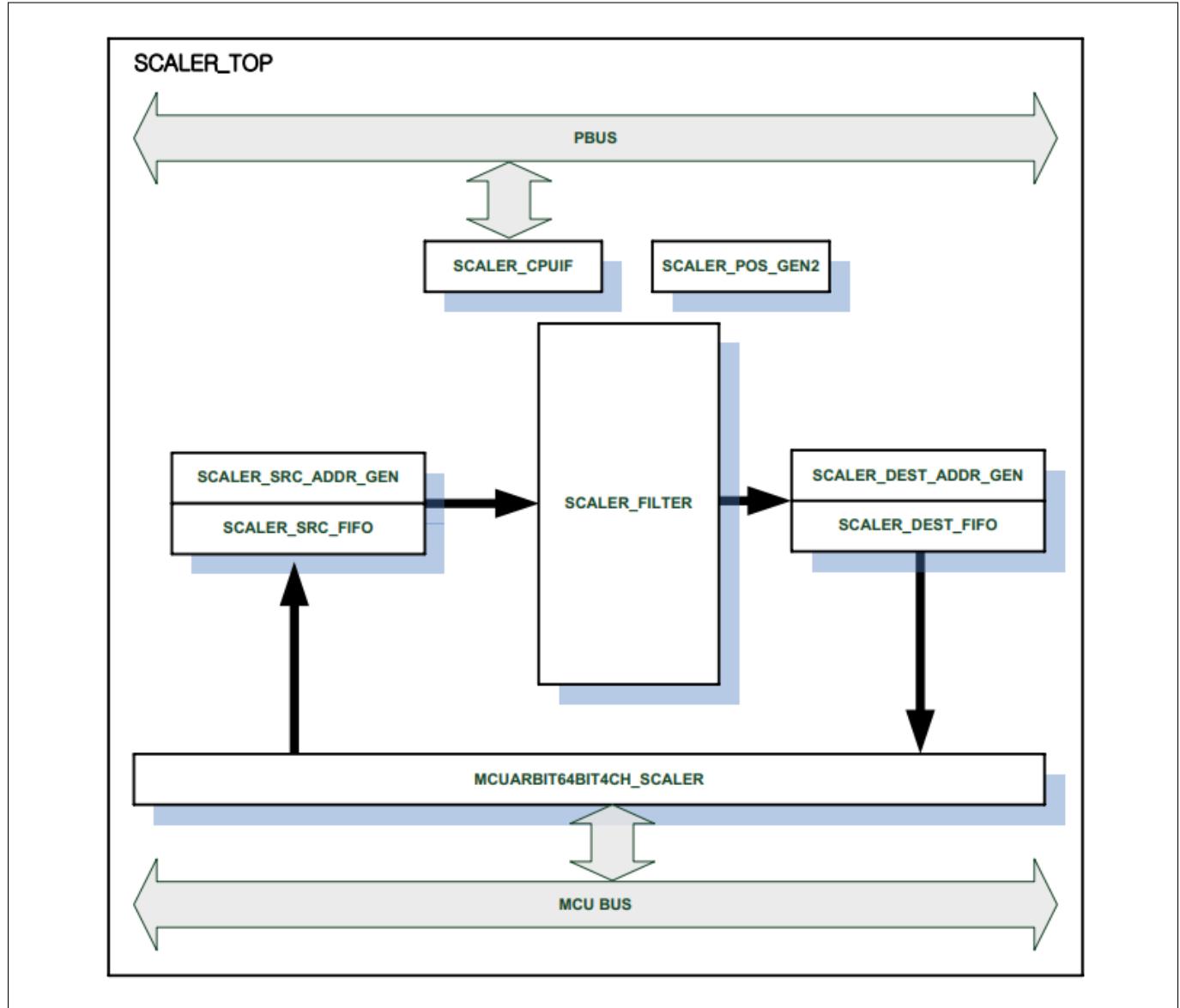


Figure 12.1: Fine Scaler Filter Block Diagram

## 12.4 Functional Description

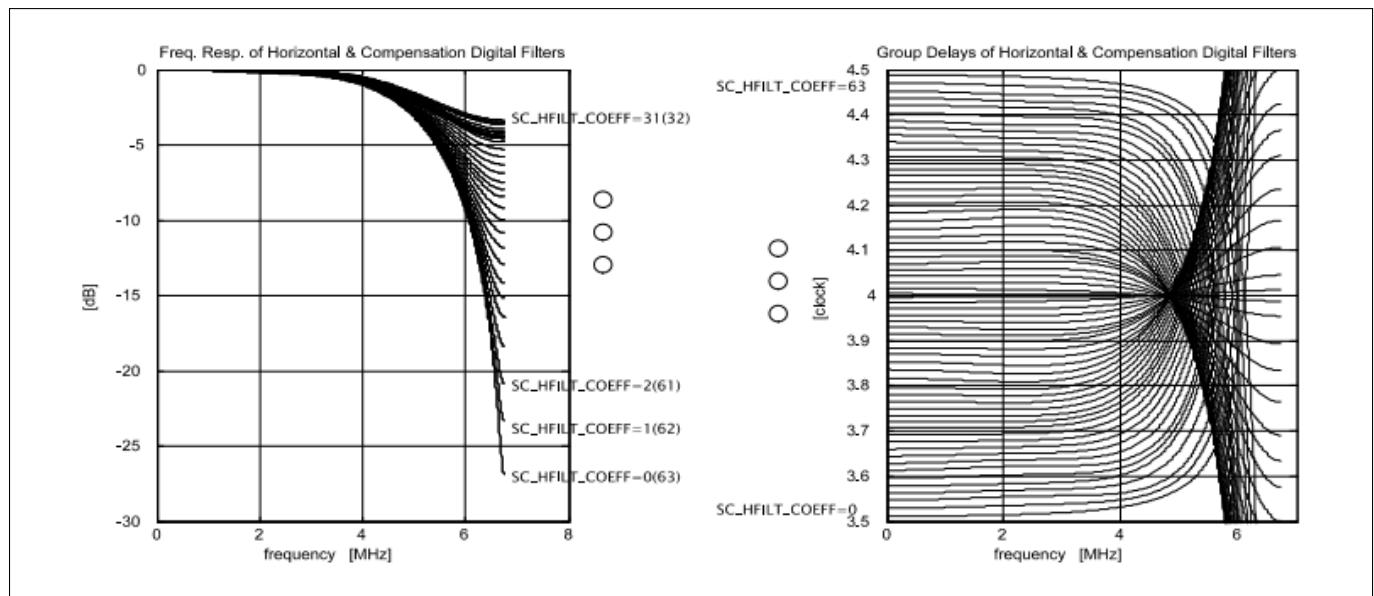
The Scales enable a user to read a source image in memory, change the image size and store the image in the memory. The Scaler changes image sizes by using the setting values of the address, width and height of the source and destination images. A user can use the low-pass filter and rotate functions of the Scaler.

### 12.4.1 Digital Filter Characteristics

The low-pass filter of the Scaler has the horizontal filter of 5-tab and the vertical filter of 3-tab. The Scaler prevents image quality deterioration when an image is enlarged by using the low-pass filter.

#### 12.4.1.1 Horizontal Filter (5-Tab FIR Filter) Frequency Response and Group Delay

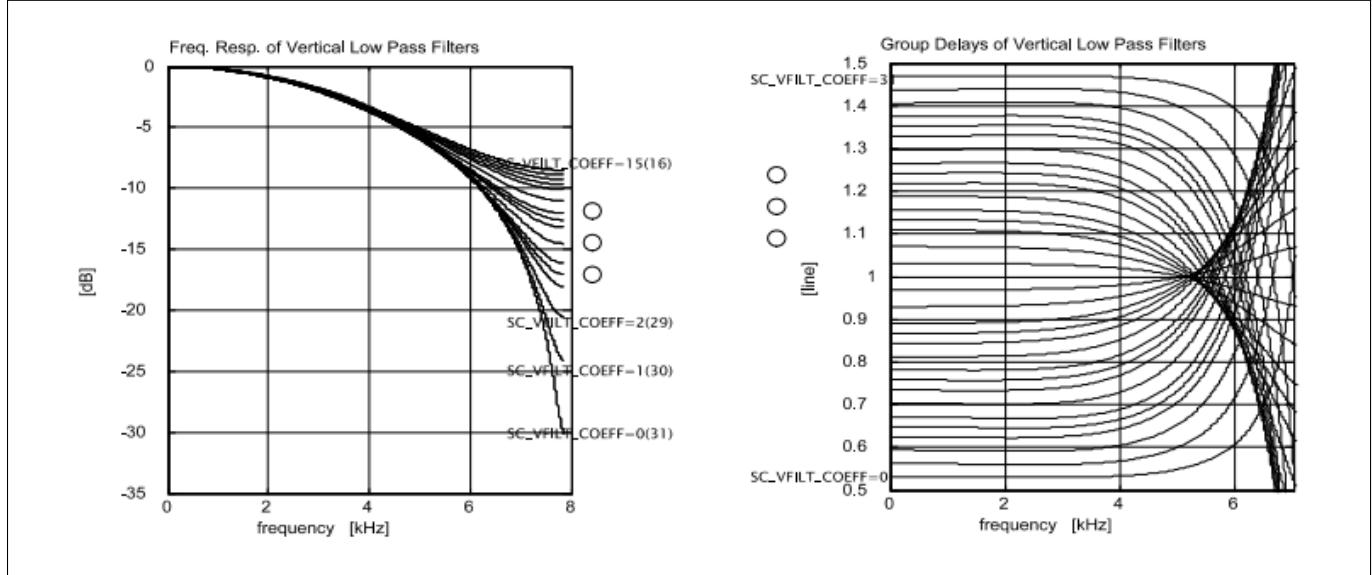
The figure below shows the characteristics of the horizontal filter of the Scaler and the filter has the setting range between 0 and 63. The Scaler register, SCCFGREG.SC\_HFILT\_COEFF, is used for the setting.



**Figure 12.2:** Horizontal Filter (5-Tab FIR Filter) Frequency Response and Group Delay

#### 12.4.1.2 Vertical Filter (3-Tab FIR Filter) Frequency Response and Group Delay

The figure below shows the characteristics of the vertical filter of Scaler and the filter has the setting range between 0 and 31. The Scaler register, SCCFGREG.SC\_VFILT\_COEFF, is used for the setting.



**Figure 12.3:** Vertical Filter (3-Tab FIR Filter) Frequency Response and Group Delay

## 12.5 Programming Guide

### 12.5.1 Configuration

1. Check the Scaler controller status: SCINTREG.SC\_BUSY = 0
2. Set the source address and stride to the SCSRCADDR and SCSRCSTRIDE register respectively.
3. Set SCSRCSIZEREG.SC\_SRC\_WIDTH and SCSRCSIZEREG.SC\_SRC\_HEIGHT register. (the width is a multiple of 8).
4. Set the destination address and stride to the SCDESTADDR0 and SCDESTSTRIDE0 register respectively. If you need to scale UV interleaved image, set the destination address and stride to the SCDESTADDR1 and SCDESTSTRIDE1 register respectively.
5. Set SCDESTSIZEREG.SC\_DEST\_WIDTH and SCDESTSIZEREG.SC\_DEST\_HEIGHT register. (the width is a multiple of 8).
6. Delta Image setting: Set DELTAXREG and DELTAYREG register.
7. Soft setting: Set Horizontal/Vertical Set HVSOFTREG register.
8. Set the filter: Set the filter as On/Off by using SCCFGREG.SC\_FILT\_ENB. Select the Filter Coefficient Set by using SCCFGREG.SC\_HFILT\_COEFF and SCCFGREG.SC\_VFILT\_COEFF.
9. Set the interrupt: SCINTREG.SC\_INT\_ENB register.

### 12.5.2 RUN

1. Set the SCRUNREG.SC\_RUN bit as "1".
2. Read the SCINTREG.SC\_BUSY register to check the operation status.
3. If the SCRUNREG.SC\_RUN bit is cleared when SCINTREG.SC\_BUSY = 1, the operation halts immediately.

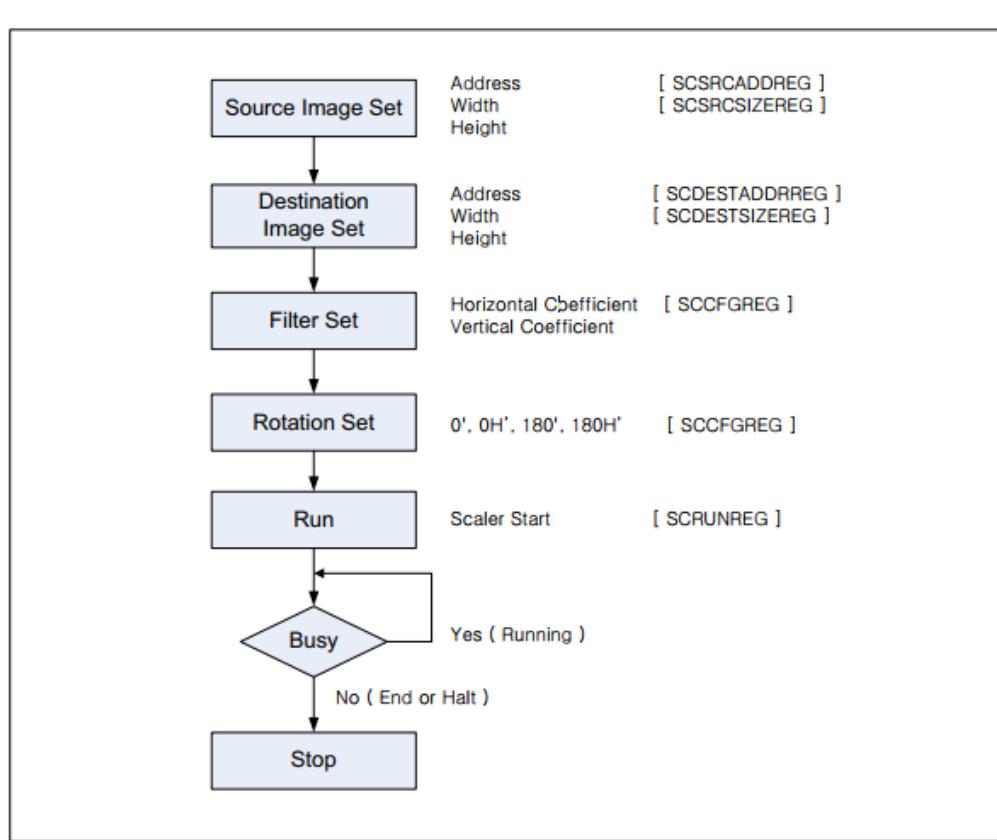


Figure 12.4: Scaler Operation Flow

## 12.6 Register Description

### 12.6.1 Register Map Summary

- Base Address: 0x2041\_0000

Register	Offset	Description	Reset Value
SCRUNREG	0x00	Scaler Run Register	0x0000_0000
SCCFGREG	0x04	Scaler Configuration Register	0x0000_0000
SCINTREG	0x08	Scaler Interrupt Register	0x0000_0000
SCSRCADDRREG	0x0C	Scaler Source Address Register	0x0000_0000
SCSRCADDRREG	0x10	Scaler Source Stride Register	0x0000_0000
SCSRCSIZEREG	0x14	Scaler Source Size Register	0x0000_0000
SCDESTADDR0	0x18	Scaler Destination Address Register 0	0x0000_0000
SCDESTSTREDE0	0x1C	Scaler Destination Stride Register 0	0x0000_0000
SCDESTADDR1	0x20	Scaler Destination Address Register 1	0x0000_0000
SCDESTSTREDE1	0x24	Scaler Destination Stride Register 1	0x0000_0000
SCDESTSIZE	0x28	Scaler Destination Size Register	0x0000_0000
DELTAXREG	0x2C	Scaler Horizontal Delta Register	0x0000_0000
DELTAYREG	0x30	Scaler Vertical Delta Register	0x0000_0000
HVSOFTRREG	0x34	Scaler Ratio Reset Value Register	0x0000_0000
CMDBUFADDR	0x38	Scaler Command Buffer Base Address Register	0x0000_0000
CMDBUFCON	0x3C	Scaler Command Buffer Control Register	0x0000_0000
YVFILTER[N]_00_03	0x40 (Filter1) 0x60 (Filter2) 0x80 (Filter3)	Scaler YV Filter[N] Value Table Register0	0x0000_0000
YVFILTER[N]_04_07	0x44 (Filter1) 0x64 (Filter2) 0x84 (Filter3)	Scaler YV Filter[N] Value Table Register1	0x0000_0000
YVFILTER[N]_08_11	0x48 (Filter1) 0x68 (Filter2) 0x88 (Filter3)	Scaler YV Filter[N] Value Table Register2	0x0000_0000
YVFILTER[N]_12_15	0x4C (Filter1) 0x6C (Filter2) 0x8C (Filter3)	Scaler YV Filter[N] Value Table Register3	0x0000_0000
YVFILTER[N]_16_19	0x50 (Filter1) 0x70 (Filter2) 0x90 (Filter3)	Scaler YV Filter[N] Value Table Register4	0x0000_0000
YVFILTER[N]_20_23	0x54 (Filter1) 0x74 (Filter2) 0x94 (Filter3)	Scaler YV Filter[N] Value Table Register5	0x0000_0000
YVFILTER[N]_24_27	0x58 (Filter1) 0x78 (Filter2) 0x98 (Filter3)	Scaler YV Filter[N] Value Table Register6	0x0000_0000
YVFILTER[N]_28_31	0x5C (Filter1) 0x7C (Filter2) 0x9C (Filter3)	Scaler YV Filter[N] Value Table Register7	0x0000_0000
RSVD	0xA0 to 0xFC	Reserved	0x0000_0000
YHFILTER[N]_00_01	0x100 (Filter1) 0x140 (Filter2) 0x180 (Filter3) 0x1C0 (Filter4) 0x200 (Filter5)	Scaler YH Filter1 to 5 Value Table Register0	0x0000_0000
YHFILTER[N]_02_03	0x104 (Filter1) 0x144 (Filter2) 0x184 (Filter3) 0x1C4 (Filter4) 0x204 (Filter5)	Scaler YH Filter1 to 5 Value Table Register1	0x0000_0000
YHFILTER[N]_04_05	0x108 (Filter1) 0x148 (Filter2) 0x188 (Filter3) 0x1C8 (Filter4) 0x208 (Filter5)	Scaler YH Filter1 to 5 Value Table Register2	0x0000_0000

YHFILTER[N]_06_07	0x10C (Filter1) 0x14C (Filter2) 0x18C (Filter3) 0x1CC (Filter4) 0x20C (Filter5)	Scaler YH Filter1 to 5 Value Table Register3	0x0000_0000
YHFILTER[N]_08_09	0x110 (Filter1) 0x150 (Filter2) 0x190 (Filter3) 0x1D0 (Filter4) 0x210 (Filter5)	Scaler YH Filter1 to 5 Value Table Register4	0x0000_0000
YHFILTER[N]_10_11	0x114 (Filter1) 0x154 (Filter2) 0x194 (Filter3) 0x1D4 (Filter4) 0x214 (Filter5)	Scaler YH Filter1 to 5 Value Table Register5	0x0000_0000
YHFILTER[N]_12_13	0x118 (Filter1) 0x158 (Filter2) 0x198 (Filter3) 0x1D8 (Filter4) 0x218 (Filter5)	Scaler YH Filter1 to 5 Value Table Register6	0x0000_0000
YHFILTER[N]_14_15	0x11C (Filter1) 0x15C (Filter2) 0x19C (Filter3) 0x1DC (Filter4) 0x21C (Filter5)	Scaler YH Filter1 to 5 Value Table Register7	0x0000_0000
YHFILTER[N]_16_17	0x120 (Filter1) 0x160 (Filter2) 0x1A0 (Filter3) 0x1E0 (Filter4) 0x220 (Filter5)	Scaler YH Filter1 to 5 Value Table Register8	0x0000_0000
YHFILTER[N]_18_19	0x124 (Filter1) 0x164 (Filter2) 0x1A4 (Filter3) 0x1E4 (Filter4) 0x224 (Filter5)	Scaler YH Filter1 to 5 Value Table Register9	0x0000_0000
YHFILTER[N]_20_21	0x128 (Filter1) 0x168 (Filter2) 0x1A8 (Filter3) 0x1E8 (Filter4) 0x228 (Filter5)	Scaler YH Filter1 to 5 Value Table Register10	0x0000_0000
YHFILTER[N]_22_23	0x12C (Filter1) 0x16C (Filter2) 0x1AC (Filter3) 0x1EC (Filter4) 0x22C (Filter5)	Scaler YH Filter1 to 5 Value Table Register11	0x0000_0000
YHFILTER[N]_24_25	0x130 (Filter1) 0x170 (Filter2) 0x1B0 (Filter3) 0x1F0 (Filter4) 0x230 (Filter5)	Scaler YH Filter1 to 5 Value Table Register12	0x0000_0000
YHFILTER[N]_26_27	0x134 (Filter1) 0x174 (Filter2) 0x1B4 (Filter3) 0x1F4 (Filter4) 0x234 (Filter5)	Scaler YH Filter1 to 5 Value Table Register13	0x0000_0000
YHFILTER[N]_28_29	0x138 (Filter1) 0x178 (Filter2) 0x1B8 (Filter3) 0x1F8 (Filter4) 0x238 (Filter5)	Scaler YH Filter1 to 5 Value Table Register14	0x0000_0000
YHFILTER[N]_30_31	0x13C (Filter1) 0x17C (Filter2) 0x1BC (Filter3) 0x1FC (Filter4) 0x23C (Filter5)	Scaler YH Filter1 to 5 Value Table Register15	0x0000_0000

### 12.6.2 SCRUNREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	0
SC_RUN	[0]	R/W	Scaler RUN bit. When the scaling process is finished, SCALER clears this bit automatically. While the scale process is running, it can be halted by clearing this bit. 0 = Stop 1 = Start (Auto Clear)	0

### 12.6.3 SCCFGREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:21]	-	Reserved	0
SC_VFILT_COEFF	[20:16]	R/W	Vertical filter coefficient select. Range is 0 to 31. (See Frequency Response Graph)	-
RSVD	[15:14]	-	Reserved	0
SC_HFILT_COEFF	[13:8]	R/W	Horizontal filter coefficient select. Range is 0 to 63. (See Frequency Response Graph)	-
RSVD	[7:2]	-	Reserved	0
SC_FILT_ENB	[1:0]	R/W	Fine scale filter enable. 00 = Filter Disable 01 = Reserved 10 = Reserved 11 = Filter Enable NOTE: This bit should be set as "00" or "11"	-

### 12.6.4 SCINTREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:25]	-	Reserved	0
SC_BUSY	[24]	R	Scaler Busy Check 0 = Scaler Idle 1 = Scaler Busy	0
RSVD	[23:18]	-	Reserved	0
CMD_PROC_INT_E_NB	[17]	R/W	Internal Command Processor Interrupt Enable 0 = Disable 1 = Enable	0
SC_INT_ENB	[16]	R/W	Scaler Interrupt Enable 0 = Disable 1 = Enable	0
RSVD	[15:10]	R/W	Reserved	0
CMD_PROC_INT_C_LR	[9]	W	Clear Internal Command Processor Interrupt Pending Bit 0 = None 1 = Clear Interrupt Pending	0
SC_INT_CLR	[8]	W	Clear Internal Command Processor Interrupt Pending Bit 0 = None 1 = Clear Interrupt Pending	0
RSVD	[7:2]	-	Reserved	0
CMD_PROC_INT_PEND	[1]	R	Command Processor Interrupt Pending Bit 0 = None 1 = Interrupt Pending	0
SC_INT_PEND	[0]	R	Scaler Interrupt Pending Bit 0 = None 1 = Interrupt Pending	0

### 12.6.5 SCSRCADDRREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SC_SRC_REG	[31:0]	R/W	Source Base Address Register	-

### 12.6.6 SCSRCADDRREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SC_SRC_STR	[31:0]	R/W	Source Stride Register	-

### 12.6.7 SCSRCSIZEREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x14 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:28]	-	Reserved	-
SC_SRC_HEIGHT	[27:16]	R/W	Source Image Height. Height's range is 8 to 4096 (Source Height - 1)	0
RSVD	[15:12]	-	Reserved	0
SC_SRC_WIDTH	[11:0]	R/W	Set Source Image Width. Width's range is 8 to 4096. Width must align to 8 (Source width - 1)	-

### 12.6.8 SCDESTADDR0

- Base Address: 0x2041\_0000
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SC_DEST_ADDR0	[31:0]	R/W	Destination Base Address0	-

### 12.6.9 SCDESTSTRIDE0

- Base Address: 0x2041\_0000
- Address = Base Address + 0x1C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SC_DEST_STRIDE0	[31:0]	R/W	Destination Stride	-

### 12.6.10 SCDESTADDR1

- Base Address: 0x2041\_0000
- Address = Base Address + 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SC_DEST_ADDR1	[31:0]	R/W	Destination Base Address1 NOTE: UV interleaved mode only	-

### 12.6.11 SCDESTSTREDE1

- Base Address: 0x2041\_0000
- Address = Base Address + 0x24 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SC_DEST_STRIDE1	[31:0]	R/W	Destination Stride NOTE: UV interleaved mode only	-

### 12.6.12 SCDESTSIZE

- Base Address: 0x2041\_0000
- Address = Base Address + 0x28 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:28]	-	Reserved	0
SC_DEST_HEIGHT	[27:16]	R/W	Destination Image Height Height's range is 8 to 4096 (Destination Height -1)	-
RSVD	[15:12]	-	Reserved	0
SC_DEST_WIDTH	[11:0]	R/W	Destination Image Width Image's range is 8 to 4096	-

### 12.6.13 DELTAXREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x2C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DELTAXREG	[31:0]	R/W	Delta X of X-axis $\text{DELTAX} = (\text{sc_src_width} \times \text{h'10000}) / (\text{sc_dest_width}-1)$	0

### 12.6.14 DELTAYREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DELTAYREG	[31:0]	R/W	Delta Y of Y-axis $\text{DELTAY} = (\text{sc_src_height} \times \text{h'10000}) / (\text{sc_dest_height}-1)$	0

### 12.6.15 HVSOFTREG

- Base Address: 0x2041\_0000
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:21]	-	Reserved	-
V_RATIO	[20:16]	R/W	Vertical Filter Ratio	0
RSVD	[15:6]	-	Reserved	-
H_RATIO	[5:0]	R/W	Horizontal Filter Ratio	0

### 12.6.16 CMDBUFADDR

- Base Address: 0x2041\_0000
- Address = Base Address + 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CMDBUFADDR	[31:0]	R/W	Scaler Command Buffer Base Address Register	-

### 12.6.17 CMDBUFCON

- Base Address: 0x2041\_0000
- Address = Base Address + 0x3C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
CMDBUF_STOP	[1]	R/W	Scaler Command Buffer Stop Register 0 = No operation 1 = Stop	0
CMDBUF_START	[0]	R/W	Scaler Command Buffer Start Register 0 = No operation 1 = Start	0

### 12.6.18 YVFILTER[N]\_00\_03

- Base Address: 0x2041\_0000
- Address = Base Address + 0x40 (Filter1), 0x60 (Filter2), 0x80 (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_03	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_02	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_01	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_00	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.19 YVFILTER[N]\_04\_07

- Base Address: 0x2041\_0000
- Address = Base Address + 0x44 (Filter1), 0x64 (Filter2), 0x84 (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_07	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_06	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_05	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_04	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.20 YVFILTER[N]\_08\_11

- Base Address: 0x2041\_0000
- Address = Base Address + 0x48 (Filter1), 0x68 (Filter2), 0x88 (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_11	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_10	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_09	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_08	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.21 YVFILTER[N]\_12\_15

- Base Address: 0x2041\_0000
- Address = Base Address + 0x4C (Filter1), 0x6C (Filter2), 0x8C (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_15	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_14	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_13	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_12	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.22 YVFILTER[N]\_16\_19

- Base Address: 0x2041\_0000
- Address = Base Address + 0x50 (Filter1), 0x70 (Filter2), 0x90 (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_19	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_18	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_17	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_17	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.23 YVFILTER[N]\_20\_23

- Base Address: 0x2041\_0000
- Address = Base Address + 0x54 (Filter1), 0x74 (Filter2), 0x94 (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_23	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_22	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_21	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_20	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.24 YVFILTER[N]\_24\_27

- Base Address: 0x2041\_0000
- Address = Base Address + 0x58 (Filter1), 0x78 (Filter2), 0x98 (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_27	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_26	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_25	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_24	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.25 YVFILTER[N]\_28\_31

- Base Address: 0x2041\_0000
- Address = Base Address + 0x5C (Filter1), 0x7C (Filter2), 0x9C (Filter3), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
FILTER_YVCOEF[N]_31	[31:24]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_30	[23:16]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_29	[15:8]	W	Scaler YVCOEF[N] value	0
FILTER_YVCOEF[N]_28	[7:0]	W	Scaler YVCOEF[N] value	0

### 12.6.26 YHFILTER[N]\_00\_01

- Base Address: 0x2041\_0000
- Address = Base Address + 0x100 (Filter1), 0x140 (Filter2), 0x180 (Filter3), 0x1C0 (Filter4), 0x200 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.27 YHFILTER[N]\_02\_03

- Base Address: 0x2041\_0000
- Address = Base Address + 0x104 (Filter1), 0x144 (Filter2), 0x184 (Filter3), 0x1C4 (Filter4), 0x204 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.28 YHFILTER[N]\_04\_05

- Base Address: 0x2041\_0000
- Address = Base Address + 0x108 (Filter1), 0x148 (Filter2), 0x188 (Filter3), 0x1C8 (Filter4), 0x208 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.29 YHFILTER[N]\_06\_07

- Base Address: 0x2041\_0000
- Address = Base Address + 0x10C (Filter1), 0x14C (Filter2), 0x18C (Filter3), 0x1CC (Filter4), 0x20C (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.30 YHFILTER[N]\_08\_09

- Base Address: 0x2041\_0000
- Address = Base Address + 0x110 (Filter1), 0x150 (Filter2), 0x190 (Filter3), 0x1D0 (Filter4), 0x210 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.31 YHFILTER[N]\_10\_11

- Base Address: 0x2041\_0000
- Address = Base Address + 0x114 (Filter1), 0x154 (Filter2), 0x194 (Filter3), 0x1D4 (Filter4), 0x214 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.32 YHFILTER[N]\_12\_13

- Base Address: 0x2041\_0000
- Address = Base Address + 0x118 (Filter1), 0x158 (Filter2), 0x198 (Filter3), 0x1D8 (Filter4), 0x218 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.33 YHFILTER[N]\_14\_15

- Base Address: 0x2041\_0000
- Address = Base Address + 0x11C (Filter1), 0x15C (Filter2), 0x19C (Filter3), 0x1DC (Filter4), 0x21C (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.34 YHFILTER[N]\_16\_17

- Base Address: 0x2041\_0000
- Address = Base Address + 0x120 (Filter1), 0x160 (Filter2), 0x1A0 (Filter3), 0x1E0 (Filter4), 0x220 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.35 YHFILTER[N]\_18\_19

- Base Address: 0x2041\_0000
- Address = Base Address + 0x124 (Filter1), 0x164 (Filter2), 0x1A4 (Filter3), 0x1E4 (Filter4), 0x224 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.36 YHFILTER[N]\_20\_21

- Base Address: 0x2041\_0000
- Address = Base Address + 0x128 (Filter1), 0x168 (Filter2), 0x1A8 (Filter3), 0x1E8 (Filter4), 0x228 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.37 YHFILTER[N]\_22\_23

- Base Address: 0x2041\_0000
- Address = Base Address + 0x12C (Filter1), 0x16C (Filter2), 0x1AC (Filter3), 0x1EC (Filter4), 0x22C (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.38 YHFILTER[N]\_24\_25

- Base Address: 0x2041\_0000
- Address = Base Address + 0x130 (Filter1), 0x170 (Filter2), 0x1B0 (Filter3), 0x1F0 (Filter4), 0x230 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_01	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_00	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.39 YHFILTER[N]\_26\_27

- Base Address: 0x2041\_0000
- Address = Base Address + 0x134 (Filter1), 0x174 (Filter2), 0x1B4 (Filter3), 0x1F4 (Filter4), 0x234 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_26	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_27	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.40 YHFILTER[N]\_28\_29

- Base Address: 0x2041\_0000
- Address = Base Address + 0x138 (Filter1), 0x178 (Filter2), 0x1B8 (Filter3), 0x1F8 (Filter4), 0x238 (Filter5), Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_26	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_27	[9:0]	W	Scaler YHCOEF[N] value	0

### 12.6.41 YHFILTER[N]\_30\_31

- Base Address: 0x2041\_0000
- Address = Base Address + 0x13C (Filter1), 0x17C (Filter2), 0x1BC (Filter3), 0x1FC (Filter4), 0x23C (Filter5),  
Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
FILTER_YHCOEF[N]_26	[25:16]	W	Scaler YHCOEF[N] value	0
RSVD	[15:10]	-	Reserved	-
FILTER_YHCOEF[N]_27	[9:0]	W	Scaler YHCOEF[N] value	0

# 13 Crypto Engine

## 13.1 Overview

Crypto Engine block executes AES, DES, HASH Encryption and Decryption.

## 13.2 Features

### 13.2.1 Crypto Engine

- Big-endian Encryption & Decryption
- Supports DMA Interface
- Supports AES ECB, CBC, CTR -128,192, 256 Mode
- Supports DES ECB, CBC -64 Mode
- Supports 3DES -64 Mode
- Supports HASH Mode (SHA1, MD5)
- Supports input share Mode (AES & HASH)
- Supports AES and HASH working at the same time (refer in the following figure)

### 13.3 Block Diagram

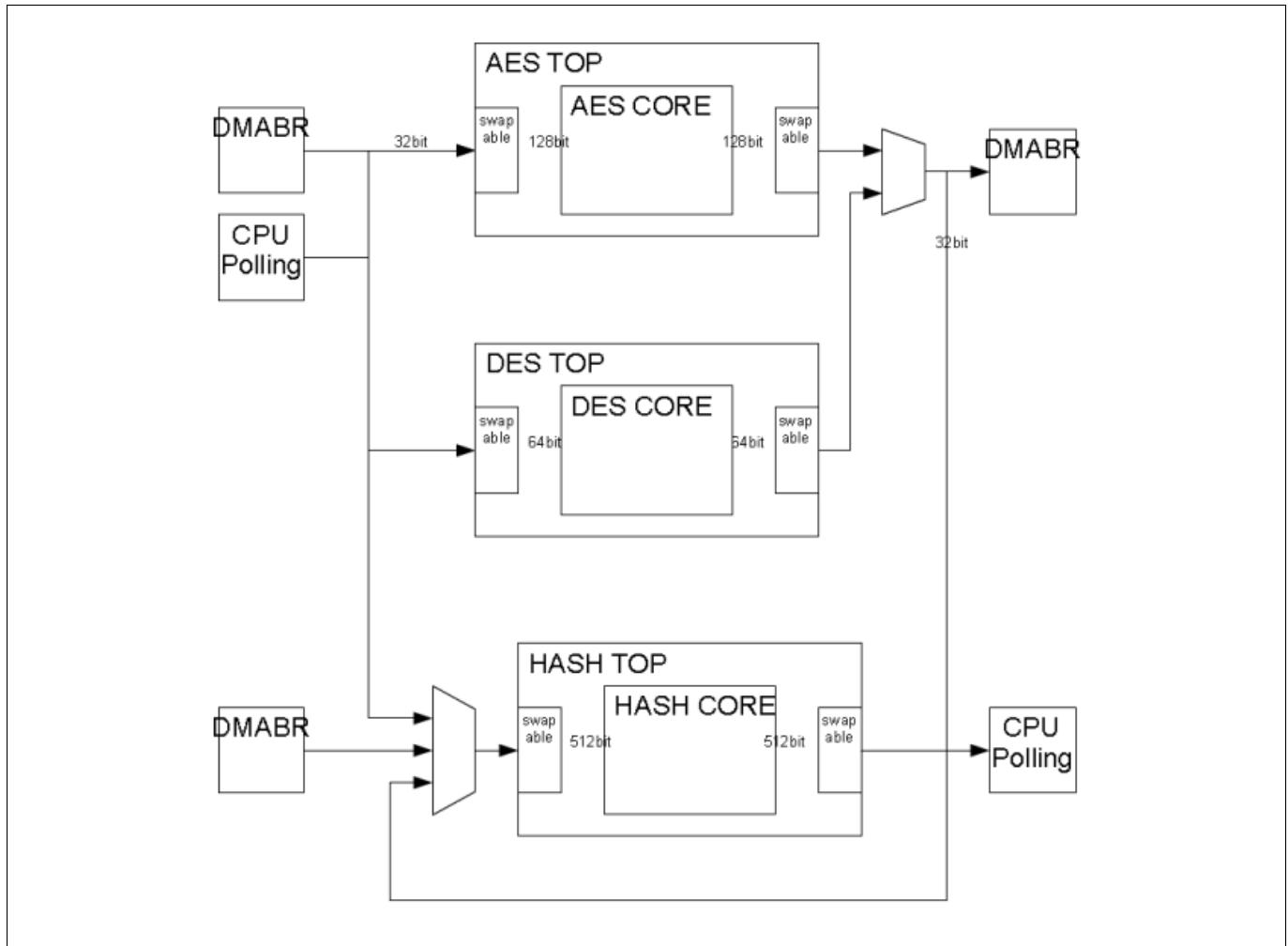


Figure 13.1: CRYPTO Block Diagram

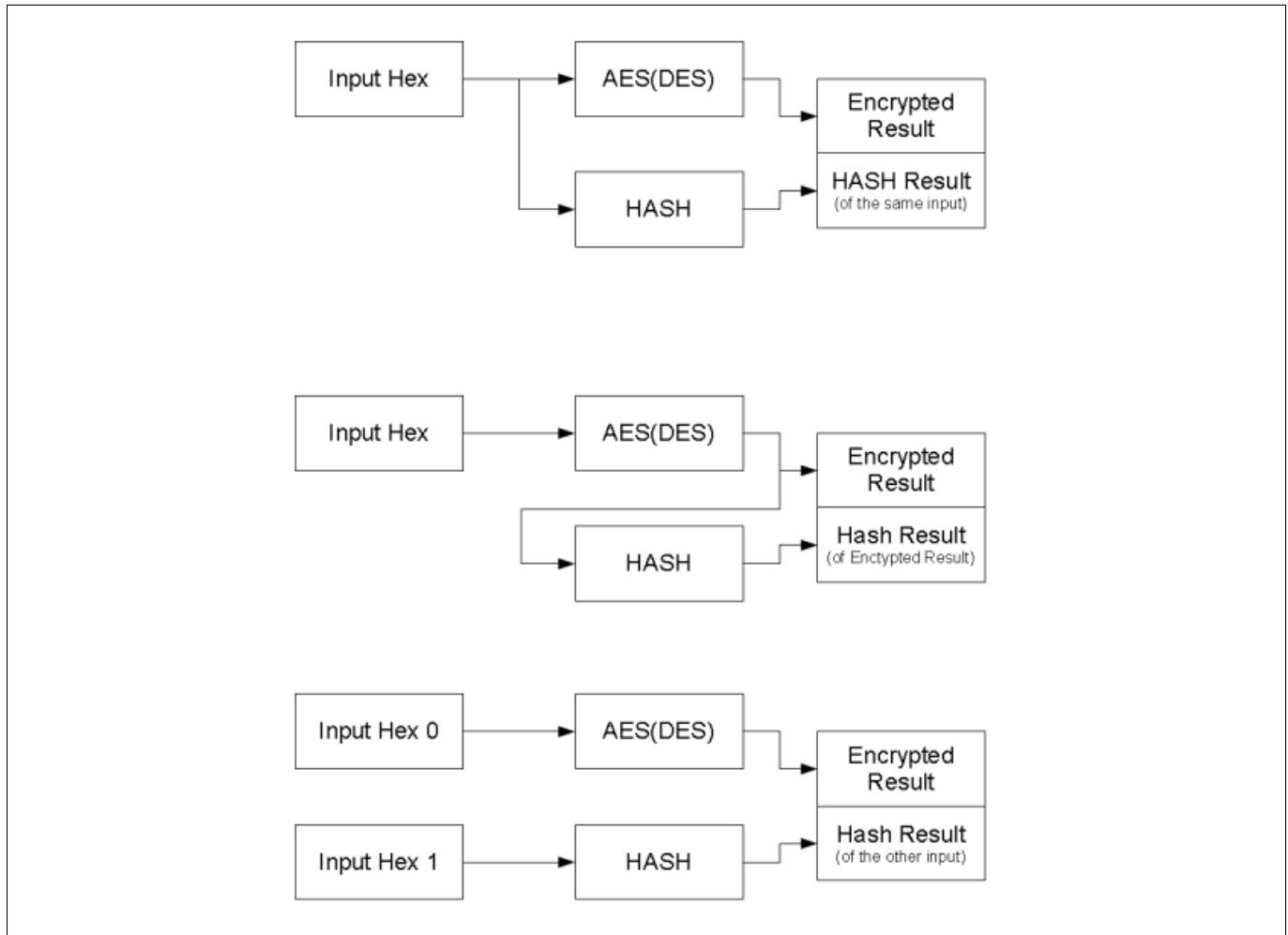


Figure 13.2: CRYPTOAES & HASH Operation Scenario

## 13.4 Functional Description

### 13.4.1 Polling Mode

In polling mode, CPU can write/read the register directly.

Access Sequence:

1. Set AES Key
2. Set Initial Vector and Set CPU\_AES\_LOADCNT with 1
3. Set AES\_TESTIN
4. Set AES Control Register & Enable
5. Set CPU\_AES\_LOADCNT with 0
6. Set IDLC\_ANNY WITH 1 (AES\_START)
7. WAIT FOR IDLE\_AES=1
8. Get Result Vector

### 13.4.2 Mode

In Channel DMA mode, users need to DMA Mode Enable.

## 13.5 Register Description

### 13.5.1 Register Map Summary

- Base Address: 0x2044\_0000

Register	Offset	Description	Reset Value
CRT_CTRL0	0x00	CRYPTO Control register	0x0000_0018
AES_CTRL0	0x0C	CRYPTO AES Control register	0x0000_0000
AES_iv0	0x10	CRYPTO AES INIT vector register 0	0x0000_0000
AES_iv1	0x14	CRYPTO AES INIT vector register 1	0x0000_0000
AES_iv2	0x18	CRYPTO AES INIT vector register 2	0x0000_0000
AES_iv3	0x1C	CRYPTO AES INIT vector register 3	0x0000_0000
AES_key0	0x30	CRYPTO AES key register 0	0x0000_0000
AES_key1	0x34	CRYPTO AES key register 1	0x0000_0000
AES_key2	0x38	CRYPTO AES key register 2	0x0000_0000
AES_key3	0x3C	CRYPTO AES key register 3	0x0000_0000
AES_key4	0x40	CRYPTO AES key register 4	0x0000_0000
AES_key5	0x44	CRYPTO AES key register 5	0x0000_0000
AES_key6	0x48	CRYPTO AES key register 6	0x0000_0000
AES_key7	0x4C	CRYPTO AES key register 7	0x0000_0000
AES_TEXTIN0	0x50	CRYPTO AES TEXTIN register 0	0x0000_0000
AES_TEXTIN1	0x54	CRYPTO AES TEXTIN register 1	0x0000_0000
AES_TEXTIN2	0x58	CRYPTO AES TEXTIN register 2	0x0000_0000
AES_TEXTIN3	0x5C	CRYPTO AES TEXTIN register 3	0x0000_0000
AES_TEXTOUT0	0x60	CRYPTO AES TEXTOUT register 0	0x0000_0000
AES_TEXTOUT1	0x64	CRYPTO AES TEXTOUT register 1	0x0000_0000
AES_TEXTOUT2	0x68	CRYPTO AES TEXTOUT register 2	0x0000_0000
AES_TEXTOUT3	0x6C	CRYPTO AES TEXTOUT register 3	0x0000_0000
DES_CTRL0	0x70	CRYPTO DES Control register	0x0000_0000
DES_iv0	0x74	CRYPTO DES INIT vector register 0	0x0000_0000
DES_iv1	0x78	CRYPTO DES INIT vector register 1	0x0000_0000
DES_KEY0_0	0x7C	CRYPTO DES KEY register 0_0	0x0000_0000
DES_KEY0_1	0x80	CRYPTO DES KEY register 0_1	0x0000_0000
DES_KEY1_0	0x84	CRYPTO DES KEY register 1_0	0x0000_0000
DES_KEY1_1	0x88	CRYPTO DES KEY register 1_1	0x0000_0000
DES_KEY2_0	0x8C	CRYPTO DES KEY register 2_0	0x0000_0000
DES_KEY2_1	0x90	CRYPTO DES KEY register 2_1	0x0000_0000
DES_TEXTIN0	0x94	CRYPTO DES TEXTIN register	0x0000_0000
DES_TEXTIN1	0x98	CRYPTO DES TEXTIN register 1	0x0000_0000
DES_TEXTOUT0	0x9C	CRYPTO DES TEXTOUT register 0	0x0000_0000
DES_TEXTOUT1	0xA0	CRYPTO DES TEXTOUT register 1	0x0000_0000
BDMAR	0xA4	CRYPTO DMA BDMAR register	0x0000_0001
BDMAW	0xA8	CRYPTO DMA BDMAW register	0x0000_0001
HDMAR	0xAC	CRYPTO DMA HDMAR register	0x0000_0001
HASH_CTRL0	0xB0	CRYPTO HASH Control register 0	0x0000_0000
HASH_iv0	0xB4	CRYPTO HASH INIT TABLE register 0	0x0000_0000
HASH_iv1	0xB8	CRYPTO HASH INIT TABLE register 1	0x0000_0000
HASH_iv2	0xBC	CRYPTO HASH INIT TABLE register 2	0x0000_0000
HASH_iv3	0xC0	CRYPTO HASH INIT TABLE register 3	0x0000_0000
HASH_iv4	0xC4	CRYPTO HASH INIT TABLE register 4	0x0000_0000
HASH_TEXTOUT0	0xC8	CRYPTO HASH TEXTOUT register 0	0x0000_0000
HASH_TEXTOUT1	0xCC	CRYPTO HASH TEXTOUT register 1	0x0000_0000
HASH_TEXTOUT2	0xD0	CRYPTO HASH TEXTOUT register 2	0x0000_0000

HASH_TEXTOUT3	0xD4	CRYPTO HASH TEXTOUT register 3	0x0000_0000
HASH_TEXTOUT4	0xD8	CRYPTO HASH TEXTOUT register 4	0x0000_0000
HASH_TEXTIN	0xDC	CRYPTO HASH TEXTIN register 0	0x0000_0000
HASH_MSG_SIZE	0xE0	CRYPTO HASH TMSG SIZE register 0	0x0000_0000
HASH_MSG_SIZE	0xE4	CRYPTO HASH TMSG SIZE register 1	0x0000_0000

### 13.5.2 CRT\_CTRL0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x00 Reset Value = 0x0000\_0018

Name	Bit	Type	Description	Reset Value
RSVD	[31:11]	R	Reserved	0
CPU_INT_ENB	[10]	R/W	Enable the Interrupt 0 = Disable 1 = Enable	0
CPU_INT_MASK	[9]	R/W	Masking the Interrupt 0 = Masked 1 = Not Masked	0
CPU_DMAW_SRC	[8]	R/W	Specifies the DMA En/Decryption Mode 0 = AES 1 = DES	0
IDLE_HASHCORE	[7]	R	Indicates the HASH CORE is Idle 0 = Not Idle 1 = Idle	0
RSVD	[6]	R	Reserved	0
INTPEND	[5]	R/W	Read: Interrupt Pending Bit Write 1: Interrupt Pending Clear	0
CPU_DES_LOADCNT	[4]	W	Users must this bit to 1 after users set the DES Initial Value.	-
CPU_AES_LOADCNT	[3]	W	Users must this bit to 1 after users set the AES Initial Value.	-
IDLE_HASH	[2]	R/W	Indicates the HASH is Idle Write 1: HASH Start	0
IDLE_DES	[1]	R/W	Indicates the DES is Idle Write 1: DES Start	0
IDLE_AES	[0]	R/W	Indicates the AES is Idle Write 1: AES Start	0

### 13.5.3 AES\_CTRL0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	R	Reserved	0
AES_SELKEY	[15]	R/W	Select the AES KEY Mode 0 = CPU configuration 1 = ECID AESKEY	0
RSVD	[14:10]	R	Reserved	0
AES_SWAPOUT	[9]	R/W	Enable the AES Output Swap 0 = Not Swap 1 = Masked	0
AES_SWAPIN	[8]	R/W	Enable the AES Input Swap 0 = Not Swap 1 = Masked	0
AES_BLKMODE	[7:6]	R/W	Specifies the AES Block Mode 0 = ECB 1 = CBC 2 = CTR 3 = Reserved	0
AES_MODE	[5:4]	R/W	Specifies the AES bit Mode 0 = 128-bit 1 = 192-bit 2 = 256-bit 3 = Reserved	0
AES_128CNT	[3]	R/W	Enable the AES 128-bit Counter	0
AES_DMAMODE	[2]	R/W	Enable the AES DMA Interface 0 = Disable 1 = Enable	0
AES_ENC	[1]	R/W	Specifies the AES Encoding Mode (Encryption/Decryption) 0 = Decryption 1 = Modulation	0
AES_ENB	[0]	R/W	Enable the AES Mode 0 = Disable 1 = Enable	0

### 13.5.4 AES\_iv0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_IV	[31:0]	R/W	O_CPU_AES_IV[127:96]. AES Initial vector	0

### 13.5.5 AES\_iv1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x14 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_IV	[31:0]	R/W	O_CPU_AES_IV[95:64]. AES Initial vector	0

### 13.5.6 AES\_iv2

- Base Address: 0x2044\_0000
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_IV	[31:0]	R/W	O_CPU_AES_IV[63:32]. AES Initial vector	0

### 13.5.7 AES\_iv3

- Base Address: 0x2044\_0000
- Address = Base Address + 0x1C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_IV	[31:0]	R/W	O_CPU_AES_IV[31:0]. AES Initial vector	0

### 13.5.8 AES\_key0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[255:224]. AES Key (AES_SELKEY = 0)	0

### 13.5.9 AES\_key1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[223:192]. AES Key (AES_SELKEY = 0)	0

### 13.5.10 AES\_key2

- Base Address: 0x2044\_0000
- Address = Base Address + 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[191:160]. AES Key (AES_SELKEY = 0)	0

### 13.5.11 AES\_key3

- Base Address: 0x2044\_0000
- Address = Base Address + 0x3C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[159:128]. AES Key (AES_SELKEY = 0)	0

### 13.5.12 AES\_key4

- Base Address: 0x2044\_0000
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[127:96]. AES Key (AES_SELKEY = 0)	0

### 13.5.13 AES\_key5

- Base Address: 0x2044\_0000
- Address = Base Address + 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[95:64]. AES Key (AES_SELKEY = 0)	0

### 13.5.14 AES\_key6

- Base Address: 0x2044\_0000
- Address = Base Address + 0x48 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[63:32]. AES Key (AES_SELKEY = 0)	0

### 13.5.15 AES\_key7

- Base Address: 0x2044\_0000
- Address = Base Address + 0x4C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_AES_key	[31:0]	R/W	O_CPU_AES_key[31:0]. AES Key (AES_SELKEY = 0)	0

### 13.5.16 AES\_TEXTIN0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTIN	[31:0]	R/W	CPU_AES_TESTIN[127:96]. AES Input Vector in PIO mode (Not DMA Mode)	0

### 13.5.17 AES\_TEXTIN1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x54 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTIN	[31:0]	R/W	CPU_AES_TESTIN[95:64]. AES Input Vector in PIO mode (Not DMA Mode)	0

### 13.5.18 AES\_TEXTIN2

- Base Address: 0x2044\_0000
- Address = Base Address + 0x58 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTIN	[31:0]	R/W	CPU_AES_TESTIN[63:32]. AES Input Vector in PIO mode (Not DMA Mode)	0

### 13.5.19 AES\_TEXTIN3

- Base Address: 0x2044\_0000
- Address = Base Address + 0x5C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTIN	[31:0]	R/W	CPU_AES_TESTIN[31:0]. AES Input Vector in PIO mode (Not DMA Mode)	0

### 13.5.20 AES\_TEXTOUT0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x60 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTOUT	[31:0]	R/W	CPU_AES_TESTOUT[127:96]. AES Result Vector in PIO mode (Not DMA Mode)	0

### 13.5.21 AES\_TEXTOUT1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x64 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTOUT	[31:0]	R/W	CPU_AES_TESTOUT[95:64]. AES Result Vector in PIO mode (Not DMA Mode)	0

### 13.5.22 AES\_TEXTOUT2

- Base Address: 0x2044\_0000
- Address = Base Address + 0x68 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTOUT	[31:0]	R/W	CPU_AES_TESTOUT[63:32]. AES Result Vector in PIO mode (Not DMA Mode)	0

### 13.5.23 AES\_TEXTOUT3

- Base Address: 0x2044\_0000
- Address = Base Address + 0x6C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_AES_TEXTOUT	[31:0]	R/W	CPU_AES_TESTOUT[31:0]. AES Result Vector in PIO mode (Not DMA Mode)	0

### 13.5.24 DES\_CTRL0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x70 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:11]	R	Reserved	0
DES_TMODE	[10:8]	R/W	3DES Mode Setting [8]: 1st stage Mode 0 = Decoding 1 = Encoding [9]: 2nd stage Mode 0 = Decoding 1 = Encoding [10]: 3th stage Mode 0 = Decoding 1 = Encoding	0
RSVD	[7]	R	Reserved	0
DES_SWAPOUT	[6]	R/W	Enable the DES Output Swap 0 = Not Swap 1 = Masked	0
DES_SWAPIN	[5]	R/W	Enable the DES Input Swap 0 = Not Swap 1 = Masked	0
DES_BLKMODE	[4]	R/W	Specifies the DES Block Mode 0 = ECB 1 = CBC	0
DES_DMAMODE	[3]	R/W	Enable the DES DMA Interface 0 = Disable 1 = Enable	0
DES_MODE	[2]	R/W	Specifies the DES Mode 0 = DES 1 = 3DES	0
DES_ENC	[1]	R/W	Specifies the DES Encoding Mode (Encryption/Decryption) 0 = Decryption 1 = Modulation	0
DES_ENB	[0]	R/W	Enable the DES Mode 0 = Disable 1 = Enable	0

### 13.5.25 DES\_iv0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x74 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_IV	[31:0]	R/W	O_CPU_DES_IV[63:32]. DES Initial vector	0

### 13.5.26 DES\_iv1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x78 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_IV	[31:0]	R/W	O_CPU_DES_IV[31:0]. DES Initial vector	0

### 13.5.27 DES\_KEY0\_0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x7C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_KEY0	[31:0]	R/W	O_CPU_DES_KEY0[63:32]. DES Key (DES and 1st stage of 3DES)	0

### 13.5.28 DES\_KEY0\_1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x80 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_KEY0	[31:0]	R/W	O_CPU_DES_KEY0[31:0]. DES Key (DES and 1st stage of 3DES)	0

### 13.5.29 DES\_KEY1\_0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x84 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_KEY0	[31:0]	R/W	O_CPU_DES_KEY1[63:32]. DES Key (2nd stage of 3DES)	0

### 13.5.30 DES\_KEY1\_1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x88 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_KEY0	[31:0]	R/W	O_CPU_DES_KEY1[31:0]. DES Key (2nd stage of 3DES)	0

### 13.5.31 DES\_KEY2\_0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x8C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_KEY0	[31:0]	R/W	O_CPU_DES_KEY2[63:32]. DES Key (3th stage of 3DES)	0

### 13.5.32 DES\_KEY2\_1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x90 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_DES_KEY0	[31:0]	R/W	O_CPU_DES_KEY2[31:0]. DES Key (3th stage of 3DES)	0

### 13.5.33 DES\_TEXTIN0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x94 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_DES_TESTIN	[31:0]	R/W	CPU_DES_TESTIN[63:32]. DES Input vector in PIO mode (Not DMA Mode)	0

### 13.5.34 DES\_TEXTIN1

- Base Address: 0x2044\_0000
- Address = Base Address + 0x98 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_DES_TESTIN	[31:0]	R/W	CPU_DES_TESTIN[31:0]. DES Input vector in PIO mode (Not DMA Mode)	0

### 13.5.35 DES\_TEXTOUT0

- Base Address: 0x2044\_0000
- Address = Base Address + 0x9C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_DES_TESTOUT	[31:0]	R/W	CPU_DES_TESTOUT[63:32]. DES Input vector in PIO mode (Not DMA Mode)	0

### 13.5.36 DES\_TEXTOUT1

- Base Address: 0x2044\_0000
- Address = Base Address + 0xA0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_DES_TESTOUT	[31:0]	R/W	CPU_DES_TESTOUT [31:0]. DES Input vector in PIO mode (Not DMA Mode)	0

### 13.5.37 BDMAR

- Base Address: 0x2044\_0000
- Address = Base Address + 0xA4 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
REG_CRT_BDMAR	[31:0]	W	DMA Access register for AES, DES Input Vectors.	-

### 13.5.38 BDMAW

- Base Address: 0x2044\_0000
- Address = Base Address + 0xA8 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
REG_CRT_BDMAW	[31:0]	W	DMA Access register for AES, DES Output Vectors. (Result Vector)	-

### 13.5.39 HDMAR

- Base Address: 0x2044\_0000
- Address = Base Address + 0xAC Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
REG_CRT_BDMAR	[31:0]	W	DMA Access register for HASH Input Vectors	-

### 13.5.40 HASH\_CTRL0

- Base Address: 0x2044\_0000
- Address = Base Address + 0xB0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:7]	R	Reserved	0
HASH_INSRC	[6:5]	R/W	Specifies the Input of HASH 0 = AES input share 1 = DES input share 2 = HRDMA 3 = BWDMA	0
HASH_SWAPIN	[4]	R/W	Enable the HASH Input Swap 0 = Not Swap 1 = Masked	0
HASH_MODE	[3]	R/W	Specifies the HASH Mode 0 = SHA1 1 = MD5	0
RSVD	[2]	R	Reserved	0
HASH_DMAMODE	[1]	R/W	Enable the HASH DMA Interface 0 = Disable 1 = Enable	0
HASH_ENB	[0]	R/W	Enable the HASH Mode 0 = Disable 1 = Enable	0

### 13.5.41 HASH\_iv0

- Base Address: 0x2044\_0000
- Address = Base Address + 0xB4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_HASH_IV	[31:0]	R/W	O_CPU_HASH_IV[159:128]. HASH Initial table	0

### 13.5.42 HASH\_iv1

- Base Address: 0x2044\_0000
- Address = Base Address + 0xB8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_HASH_IV	[31:0]	R/W	O_CPU_HASH_IV[127:96]. HASH Initial table	0

### 13.5.43 HASH\_iv2

- Base Address: 0x2044\_0000
- Address = Base Address + 0xBC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_HASH_IV	[31:0]	R/W	O_CPU_HASH_IV[95:64]. HASH Initial table	0

### 13.5.44 HASH\_iv3

- Base Address: 0x2044\_0000
- Address = Base Address + 0xC0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_HASH_IV	[31:0]	R/W	O_CPU_HASH_IV[63:32]. HASH Initial table	0

### 13.5.45 HASH\_iv4

- Base Address: 0x2044\_0000
- Address = Base Address + 0xC4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
O_CPU_HASH_IV	[31:0]	R/W	O_CPU_HASH_IV[31:0]. HASH Initial table	0

### 13.5.46 HASH\_TEXTOUT0

- Base Address: 0x2044\_0000
- Address = Base Address + 0xC8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_TEXTOUT	[31:0]	R/W	CPU_HASH_TESTOUT[159:128]. HASH result output	0

### 13.5.47 HASH\_TEXTOUT1

- Base Address: 0x2044\_0000
- Address = Base Address + 0xCC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_TEXTOUT	[31:0]	R/W	CPU_HASH_TESTOUT[127:96]. HASH result output	0

### 13.5.48 HASH\_TEXTOUT2

- Base Address: 0x2044\_0000
- Address = Base Address + 0xD0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_TEXTOUT	[31:0]	R/W	CPU_HASH_TESTOUT[96:64]. HASH result output	0

### 13.5.49 HASH\_TEXTOUT3

- Base Address: 0x2044\_0000
- Address = Base Address + 0xD4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_TEXTOUT	[31:0]	R/W	CPU_HASH_TESTOUT[63:32]. HASH result output	0

### 13.5.50 HASH\_TEXTOUT4

- Base Address: 0x2044\_0000
- Address = Base Address + 0xD8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_TEXTOUT	[31:0]	R/W	CPU_HASH_TESTOUT[31:0]. HASH result output	0

### 13.5.51 HASH\_TEXTIN

- Base Address: 0x2044\_0000
- Address = Base Address + 0xDC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_TEXTOUT	[31:0]	R/W	CPU_HASH_TESTIN[127:0]. For DMA and PIO mode	0

### 13.5.52 HASH\_MSG\_SIZE

- Base Address: 0x2044\_0000
- Address = Base Address + 0xE0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_MSGSIZE	[31:0]	R/W	CPU_HASH_MSGSIZE[63:32]. HASH result output	0

### 13.5.53 HASH\_MSG\_SIZE

- Base Address: 0x2044\_0000
- Address = Base Address + 0xE4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
CPU_HASH_MSGSIZE	[31:0]	R/W	CPU_HASH_MSGSIZE[31:0]. HASH result output	0

# 14 ADC

## 14.1 Overview

The L7GAD10B001M IP is a successive-approximation-register (SAR) type analog-to-digital converter (ADC), which consists of an 8-channel input MUX, a digital-to-analog converter (DAC), a comparator, and control logic including a SAR. The L7GAD10B001M, operating with 3.3V analog power supply and 1.2V digital power supply, consumes 3.3mW power at 1-MS/s sampling rate.

## 14.2 Functional Description

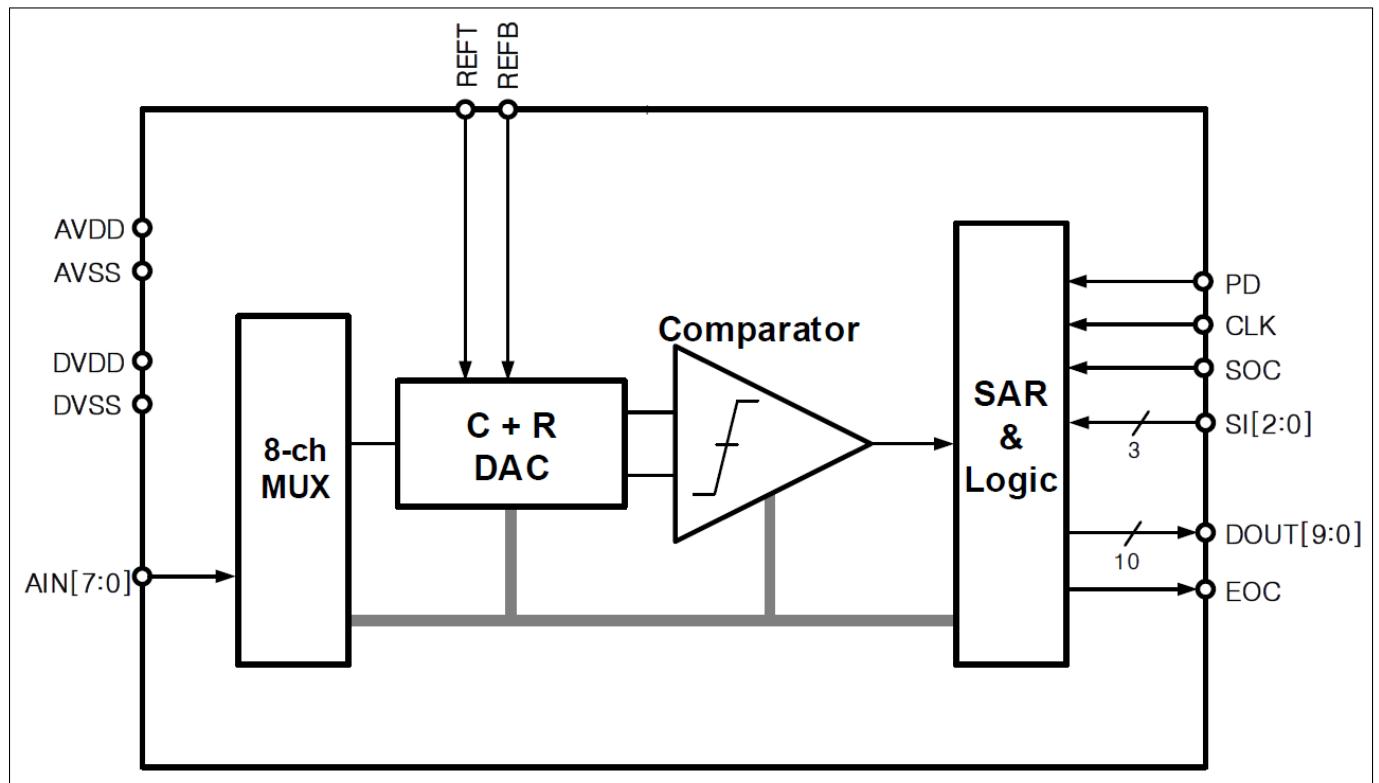


Figure 14.1: Functional Diagram

Figure 14.1 shows a functional diagram of the L7GAD10B001M. The DAC is the hybrid type incorporating a charge redistribution CDAC and RDAC combination. The input range of the ADC is limited to between the top-level reference voltage(REFT) and the bottom-level reference voltage(REFB). REFT and REFB are supplied by external reference voltage sources or analog power-andground voltage sources.

Considering the sampling rate of 1MS/s, with the 50-kHz input signal and the 3.3Vp-p input range, the L7GAD10B001M

features dynamic performance of the 76dB SFDR, 62dB SNR and 9.9-bit ENOB, and a static performance of +-1.0LSB DNL and +-2.0LSB INL. Moreover, the L7GAD10B001M consumes power-down current of less than 1uA.

## 14.3 Features

- Globalfoundries 55LP<sub>E</sub> CMOS technology
- 10-bit resolution
- 0.1MS/s to 1MS/s conversion rate
- 3.0V to 3.6V analog power supply
- 1.08V to 1.32V digital power supply or 0.95V to 1.05V digital power supply
- 8-channel single-ended inputs
- Analog input range : REFB to REFT
- Current consumption : 1.0mA (typ.)
- DNL / INL : +-1.0 /+-2.0 LSB
- SFDR : 76dB (typ.)
- SNR : 62dB (typ.)
- ENOB : 9.9-bit (typ.)
- Power-down current : < 1uA (typ.)
- Core Size : 0.09mm<sup>2</sup> (300um x 300um)

## 14.4 Connection Circuit

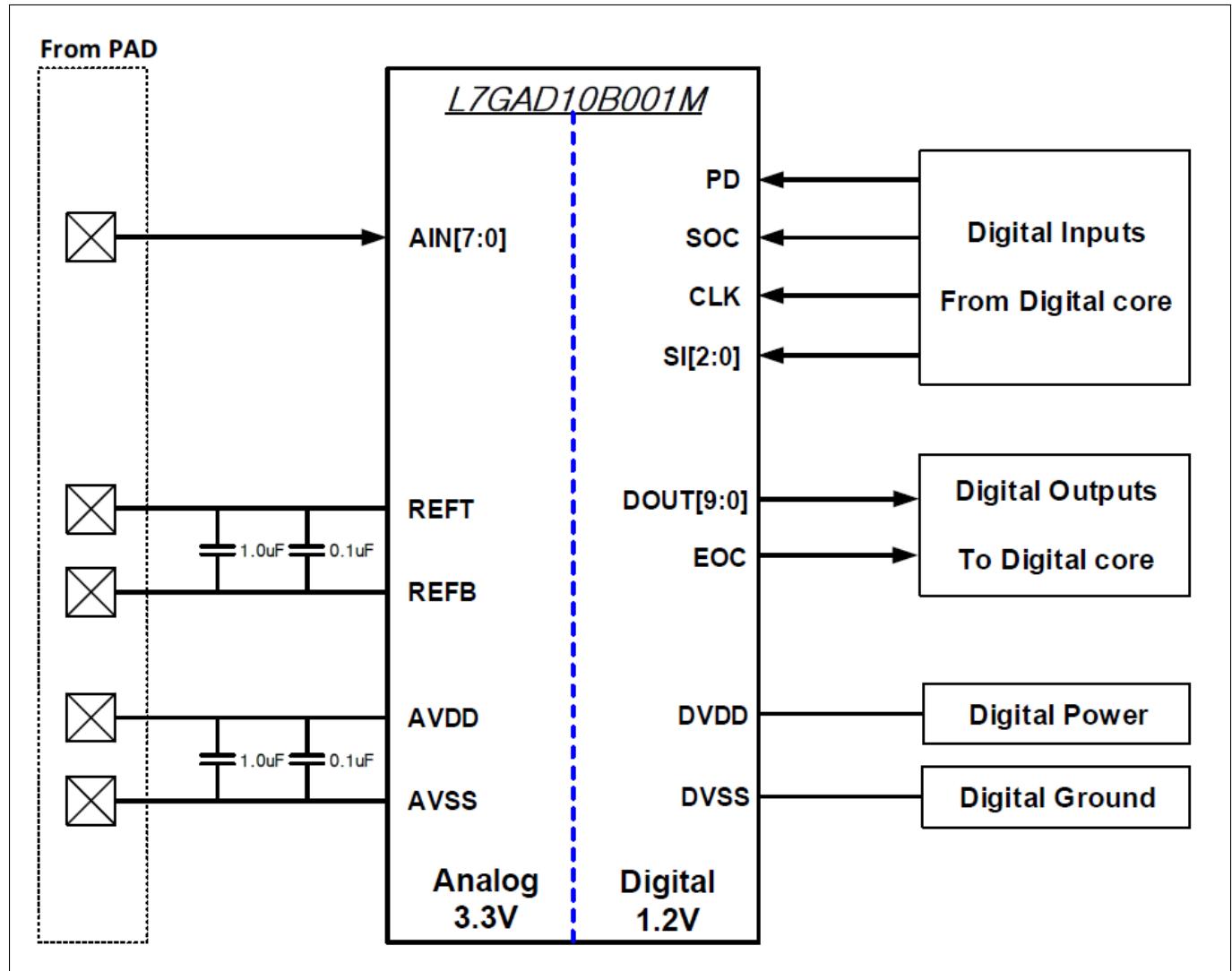


Figure 14.2: Connection Circuit

## 14.5 Timing Characteristics

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNIT
<b>TIMING SPECIFICATIONS</b>						
CLK frequency	fCLK	Main clock		12		MHz
CLK period	tCLK	1/ fCLK	83.3			ns
Conversion Cycle time	tCYCLE	$t_{CYCLE} = t_{CLK} \times 12$	1			us
Input tracking time	tTRACK	$t_{TRACK} = t_{CLK} \times 1.25$	105			ns
PD Falling to SOC Rising	t1	Analog stabilization time	2			us
SI setup time	t2		10			ns
SI hold time	t3		10			ns
SOC Falling to CLK Rising	t4		20			ns
12-CLK Rising to SOC Rising	t5		20			ns
Pulse duration of SOC high	tSOC		40			ns
EOC valid time	tEOC	$EOC = t_{CLK}$	83.3			ns
Pulse duration of PD high	tPD		1			us
Pulse duration of CLK low	tCLKL	$t_{CLKL} = t_{CLK} \times 0.5$	41.5			ns
Pulse duration of CLK high	tCLKH	$t_{CLKH} = t_{CLK} \times 0.5$	41.5			ns
12-CLK rising to DATA out delay	tD1		2			ns
11-CLK falling to EOC rising	tD2		1			ns

Table 14.1: Timing Characteristics

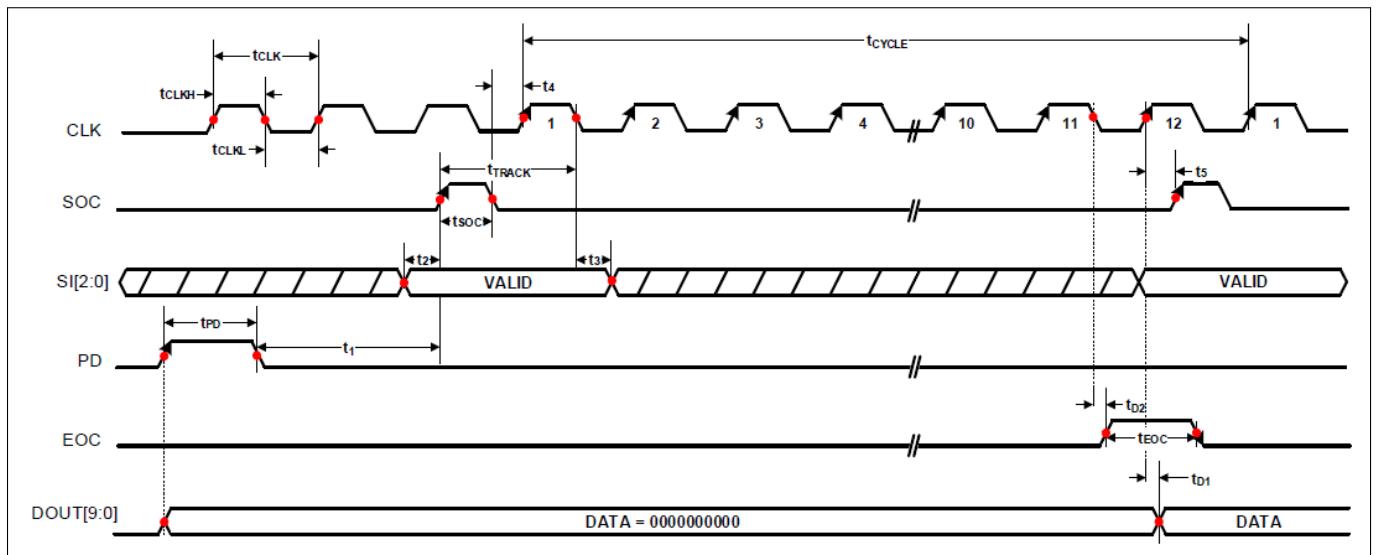


Figure 14.3: Timing Diagram

## 14.6 Timing Characteristics (Case I)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNIT
<b>TIMING SPECIFICATIONS</b>						
CLK frequency	fCLK	Main clock		14		MHz
CLK period	tCLK	1/ fCLK	71.5			ns
Conversion Cycle time	tCYCLE	$t_{CYCLE} = t_{CLK} \times 14$	1			us
Input tracking time	tTRACK	SOC Rising to 1-CLK Falling				ns
PD Falling to SOC Rising	t1	Analog stabilization time	2			us
SI setup time	t2		10			ns
SI hold time	t3		10			ns
SOC Falling to CLK Rising*	t4		20			ns
14-CLK Rising to SOC Rising	t5		-120.0			ns
Pulse duration of SOC high	tSOC		71.5			ns
EOC valid time	tEOC	$EOC = t_{CLK}$	71.5			ns
Pulse duration of PD high	tPD		1			us
Pulse duration of CLK low	tCLKL	$t_{CLKL} = t_{CLK} \times 0.5$	35.75			ns
Pulse duration of CLK high	tCLKH	$t_{CLKH} = t_{CLK} \times 0.5$	35.75			ns
12-CLK Rising to DATA out delay	tD1		2			ns
11-CLK Falling to EOC Rising	tD2		1			ns

Table 14.2: Timing Characteristics Case1

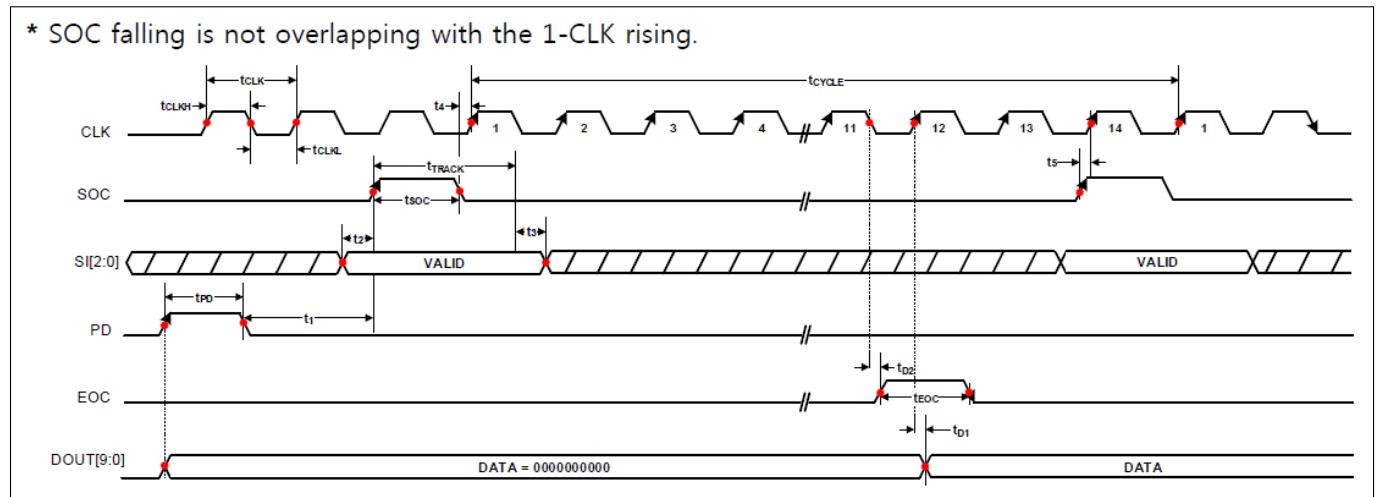


Figure 14.4: Timing Diagram

## 14.7 Timing Characteristics (Case II)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNIT
<b>TIMING SPECIFICATIONS</b>						
CLK frequency	fCLK	Main clock		14		MHz
CLK period	tCLK	1/ fCLK	71.5			ns
Conversion Cycle time	tCYCLE	$tCYCLE = tCLK \times 14$	1			us
Input tracking time	tTRACK	SOC Rising to 2-CLK Falling				ns
PD Falling to SOC Rising	t1	Analog stabilization time	2			us
SI setup time	t2		10			ns
SI hold time	t3		10			ns
CLK Rising to SOC Falling*	t4		20			ns
14-CLK Rising to SOC Rising	t5		-50.0			ns
Pulse duration of SOC high	tSOC		71.5			ns
EOC valid time	tEOC	$EOC = tCLK$	71.5			ns
Pulse duration of PD high	tPD		1			us
Pulse duration of CLK low	tCLKL	$tCLKL = tCLK \times 0.5$	35.75			ns
Pulse duration of CLK high	tCLKH	$tCLKH = tCLK \times 0.5$	35.75			ns
13-CLK rising to DATA out delay	tD1		2			ns
12-CLK falling to EOC rising	tD2		1			ns

Table 14.3: Timing Characteristics Case2

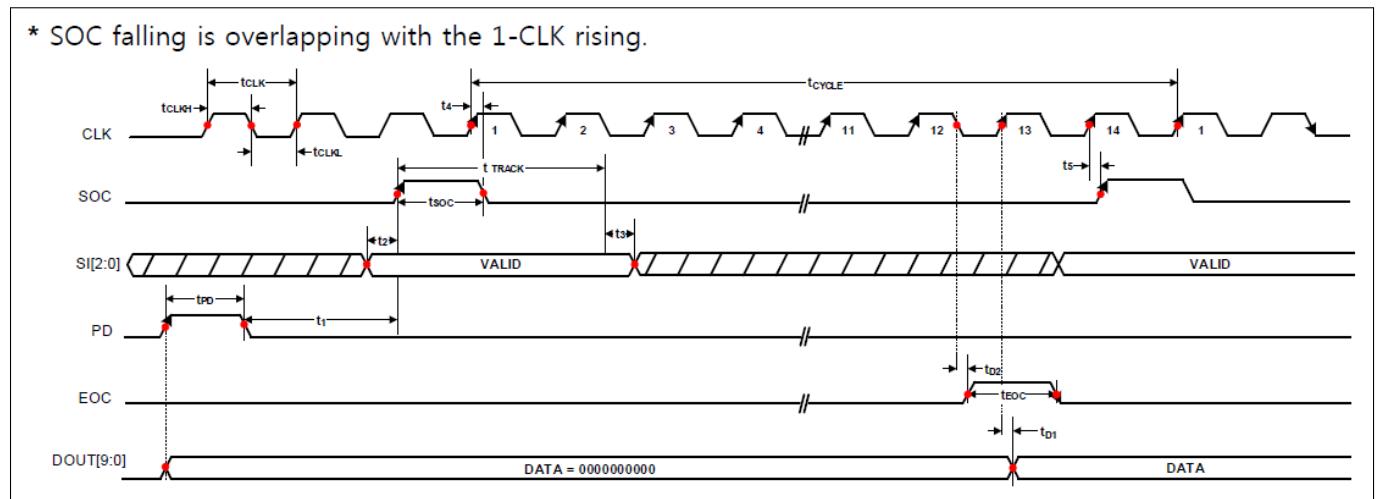


Figure 14.5: Timing Diagram

## 14.8 Pin Descriptions

PIN NAME	I/O	DESCRIPTIONS
AIN[7:0]	AI	8-channel single-ended analog inputs. AIN signal should not be crossed by any signals and should not be closely laid to digital signals. When connected to a PAD directly, these signal paths should be kept as short as possible.
REFT	AP	Top level of reference voltage. REFT port can be connected to AVDD
REFB	AG	Bottom level of reference voltage. REFB port can be connected to AVSS
PD	DI	Power-down input, active high.
SI[2:0]	DI	Selection signal of 8-channel analog inputs.
CLK	DI	Main clock input of ADC. Max conversion rate = CLK / 12
SOC	DI	Start-of-conversion signal input.
DOUT[9:0]	DO	10-bit data outputs
EOC	DO	End-of-conversion signal outputs.
AVDD	AP	Analog power.
AVSS	AG	Analog ground
DVDD	DP	Digital power
DVSS	DG	Digital ground

AI : analog input  
 DI : digital input  
 DO : digital output  
 AP : analog power  
 AG : analog ground  
 DP : digital power  
 DG : digital ground

Table 14.4: Pin Descriptions

## 14.9 ADC Register Description

### 14.9.1 Register Map Summary

- Base Address: 206C\_0000(ADC)

Register	Offset	Description	Reset Value
ADCCON	0x00	ADC CONTROL REGISTER	0x0000_0004
ADCDAT	0x04	ADC OUTPUT DATA REGISTER	0x0000_0000
ADCINTENB	0x08	ADC INTERRUPT ENABLE REGISTER	0x0000_0000
ADCINTCLR	0x0C	ADC INTERRUPT PENDING AND CLEAR REGISTER	0x0000_0000
PRESCALERCON	0x10	ADC PRESCALER REGISTER	0x0000_0000
RESERVED	0x14	RESERVED	Undefined
ADCEN	0x18	ADC ENABLE	0x0000_0001

### 14.9.2 ADCCON

- Base Address: 206C\_0000(ADC)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0004

Name	Bit	Type	Description	Reset Value
RSVD	[31:14]	-	Reserved	-
ADC_DATA_SEL	[13:10]	R/W	These bits select ADCDATA. 0000 : ADCDATA 5clk delayed by PCLK 0001 : ADCDATA 4clk delayed by PCLK 0010 : ADCDATA 3clk delayed by PCLK 0011 : ADCDATA 2clk delayed by PCLK 0100 : ADCDATA 1clk delayed by PCLK 0101 : ADCDATA which is not delayed else : ADCDATA 4clk delayed by PCLK	0
TOT_ADC_CLK_Cnt	[9:6]	R/W	These bits control the Start Of Conversion(SOC) timing. SOC signal is synchronized by (ADCCLK x TOT_ADC_CLK_Cnt).	0
asel	[5:3]	R/W	These bits select ADCIN. NXP5540 has four ADCINS and can select one of them. 000 : ADCIN_0 001 : ADCIN_1 010 : ADCIN_2 011 : ADCIN_3 100 : ADCIN_6 101 : ADCIN_7 110 : ADCIN_8 111 : ADCIN_9	0
PD	[2]	R/W	A/D Converter PowerDown Mode. If this bit is set as '0', power is actually applied to the A/D converter. 0 : ADC Power On 1 : ADC Power Off(Standby)	0
RSVD	[1]	-	Reserved	-
aden	[0]	R/W	A/D Conversion Start - When the A/D conversion ends, this bit is cleared to 0. Read > Check the A/D conversion operation. 0 : Idle 1 : Busy Write > Start the A/D conversion. 0 : None 1 : Start A/D conversion	0

### 14.9.3 ADCDAT

- Base Address: 206C\_0000(ADC)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:10]	-	Reserved	-
ADCDA	[9:0]	R	These bits are 10-bit data converted via the ADC.	0

#### 14.9.4 ADCINTENB

- Base Address: 206C\_0000(ADC)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
ADCintenb	[0]	R/W	ADC Interrupt Enable. (This bit determines the generation of an interrupt when EOC occurs.) This bit determines if interrupt occurs when the ADEN bit is 1. 0 : Interrupt Disable 1 : Interrupt Enable	0

#### 14.9.5 ADCINTCLR

- Base Address: 206C\_0000(ADC)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
adcintclr	[0]	R/W	EOC Interrupt Pending and Clear Read > 0 : None 1 : Interrupt Pended Write > 0 : None 1 : Pending Clear	0

#### 14.9.6 PRESCALERCON

- Base Address: 206C\_0000(ADC)
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
apen	[15]	R/W	Prescaler Enable.This bit determines the supply of the clock divided by the APSV register for the A/D converter.Before the APEN bit is enabled, the APSV register should be set. 0 : Disable 1 : Enable	0
RSVD	[14:10]	-	Reserved	-
PRESCALERCON	[9:0]	R/W	A/D Converter Clock Prescaler Value (10bit) - To write a value to this bit, APEN should be 1. - The maximum value of the ADC CLK divided by the APSV value is 625 kHz (1600 ns) (where PCLK is 50 MHz). - The minimum and the maximum value for the APSV bit are 19 and 255, respectively.(In effect, the range of the clock divide value is from 7 to 1024). - Input Value = Clock Divide Value -1.For divide-by-20 and divide-by-100, (20-1) = 19 and (100-1) = 99 are input to APSV, respectively.	0

### 14.9.7 ADCEN

- Base Address: 206C\_0000(ADC)
- Address = Base Address + 0x18 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
ADC_enable	[0]	R/W	Enable A/D Converter. 0 : ADC Disable 1 : ADC Enable	1

# 15 I2C

## 15.1 Product Overview

This chapter describes the DesignWare APB I2C Interface Peripheral, referred to as DW\_apb\_i2c. The DW\_apb\_i2c component is an AMBA 2.0-compliant Advanced Peripheral Bus (APB) slave device and is part of the family of DesignWare Synthesizable Components.

### 15.1.1 DesignWare System Overview

The Synopsys DesignWare Synthesizable Components environment is a parameterizable bus system containing AMBA version 2.0-compliant AHB (Advanced High-performance Bus) and APB (Advanced Peripheral Bus) components, and AMBA version 3.0-compliant AXI (Advanced eXtensible Interface) components. Figure 15.1 illustrates one example of this environment, including the AXI bus, the AHB bus, and the APB bus. Included in this subsystem are synthesizable IP for AXI/AHB/APB peripherals, bus bridges, and an AXI interconnect and AHB bus fabric. Also included are verification IP for AXI/AHB/APB master/slave models and bus monitors. In order to display the databook for a DW\_\* component, click on the corresponding component object in the illustration.

**Attention**

Links resolve only if you are viewing this databook from your \$DESIGNWARE\_HOME tree, and to only those components that are installed in the tree.

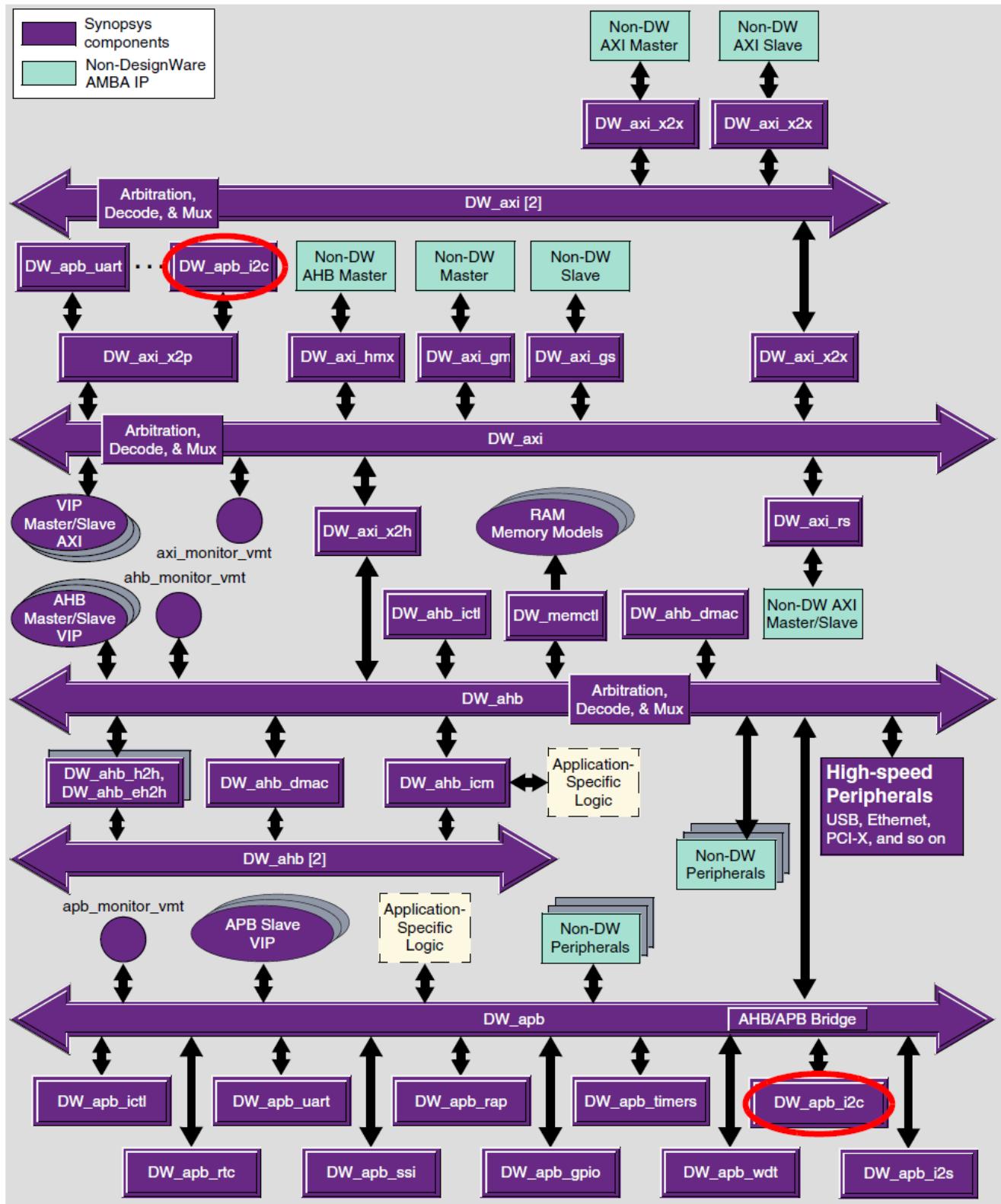


Figure 15.1: Example of DW\_apb\_i2c in a Complete System

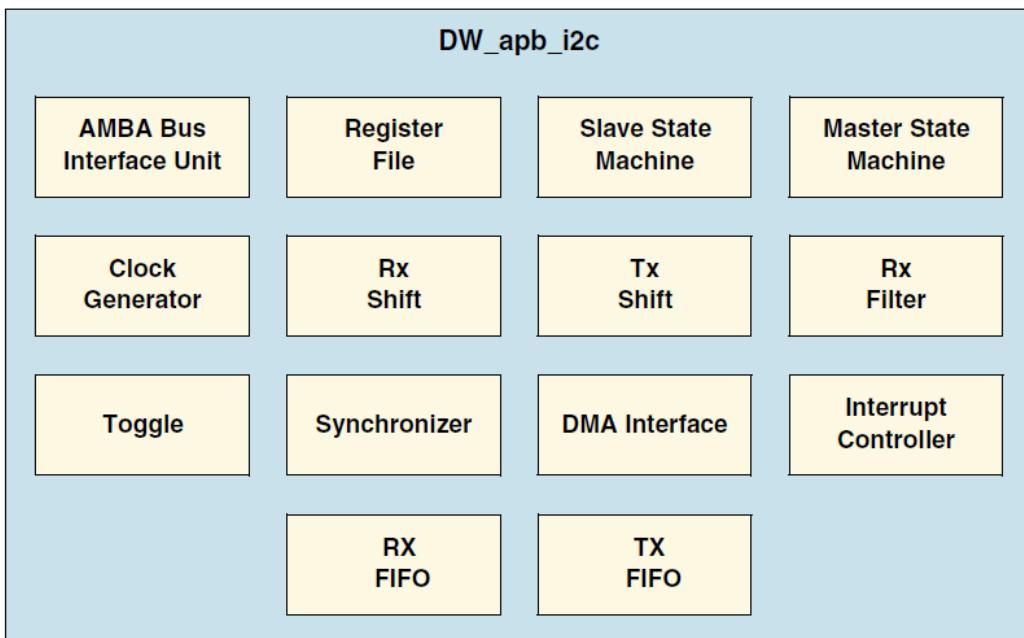
You can connect, configure, synthesize, and verify the DW\_apb\_i2c within a DesignWare subsystem using coreAssembler, documentation for which is available on the web in the coreAssembler User Guide. If you want to configure, synthesize, and verify a single component such as the DW\_apb\_i2c component, you might prefer to use coreConsultant, documentation for which is available in the coreConsultant User Guide.

### 15.1.2 General Product Description

The DW\_apb\_i2c is a configurable, synthesizable, and programmable control bus that provides support for the communications link between integrated circuits in a system. It is a simple two-wire bus with a software-defined protocol for system control, which is used in temperature sensors and voltage level translators to EEPROMs, general-purpose I/O, A/D and D/A converters, CODECs, and many types of microprocessors.

#### 15.1.2.1 DW\_apb\_i2c Block Diagram

Figure 15.2 illustrates a simple block diagram of DW\_apb\_i2c. For a more detailed block diagram and description of the component, refer to "Functional Description".



**Figure 15.2:** Block Diagram of DW\_apb\_i2c

### 15.1.3 Features

DW\_apb\_i2c has the following features:

#### 15.1.3.1 I2C Features

- Two-wire I2C serial interface which consists of a serial data line (SDA) and a serial clock (SCL)
- Three speeds:
  - Standard mode (0 to 100 Kb/s)
  - Fast mode ( $\leq$  400 Kb/s) or fast mode plus ( $\leq$  1000 Kb/s)
- Clock synchronization
- Master OR slave I2C operation
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I2C that used to share the I2C bus)
- Transmit and receive buffers
- Interrupt or polled-mode operation
- Handles Bit and Byte waiting at all bus speeds
- Simple software interface consistent with DesignWare APB peripherals
- Component parameters for configurable software driver support
- DMA handshaking interface compatible with the DW\_ahb\_dmac handshaking interface
- Programmable SDA hold time (tHD;DAT)
- Bus clear feature
- Device ID feature
- PMBus Support
- SMBus Slave detects and responds to ARP commands.

The DW\_apb\_i2c requires external hardware components as support in order to be compliant in an I2C system. The descriptions are detailed later in this document. It must also be noted that the DW\_apb\_i2c should only be operated either as (but not both):

- A master in an I2C system and programmed only as a Master; OR
- A slave in an I2C system and programmed only as a Slave.

1. In this document, references to fast mode also apply to fast mode plus, unless specifically stated otherwise.

### 15.1.3.2 DesignWare APB Slave Interface

- Support for APB data bus widths 32 bits
- Source code for this component is available on a per-project basis as a DesignWare Core; contact your local sales office for the details.

### 15.1.4 Standards Compliance

The DW\_apb\_i2c component conforms to the AMBA Specification, Revision 2.0 from ARM. Readers are assumed to be familiar with this specification. The DW\_apb\_i2c was designed for the following specifications:

- I2C Bus Specification, Version 6.0, dated April 2014
- SMBus specification Version 3.0, dated January 2015
- PMBus Specification Version 1.2, dated September 2010

## 15.2 I2C Functional Description

This chapter describes the functional behavior of DW\_apb\_i2c in more detail.

### 15.2.1 Overview

The I2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a "transmitter" or "receiver," depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

#### Note

The DW\_apb\_i2c must only be programmed to operate in either master OR slave mode only. Operating as a master and slave simultaneously is not supported.

The DW\_apb\_i2c module can operate in standard mode (with data rates 0 to 100 Kb/s), fast mode (with data rates less than or equal to 400 Kb/s), fast mode plus (with data rates less than or equal to 1000 Kb/s), high-speed mode (with data rates less than or equal to 3.4 Mb/s), and Ultra-Fast Speed Mode (with data rates less than or equal to 5 Mb/s).

#### Note

In this document, references to fast mode also apply to fast mode plus, unless specifically stated otherwise.

The DW\_apb\_i2c can communicate with devices only of these modes as long as they are attached to the bus. Additionally, high-speed mode and fast mode devices are downward compatible. For instance, high-speed mode devices can communicate with fast mode and standard mode devices in a mixed-speed bus system; fast mode devices can communicate with standard mode devices in 0 to 100 Kb/s I2C bus system. However:

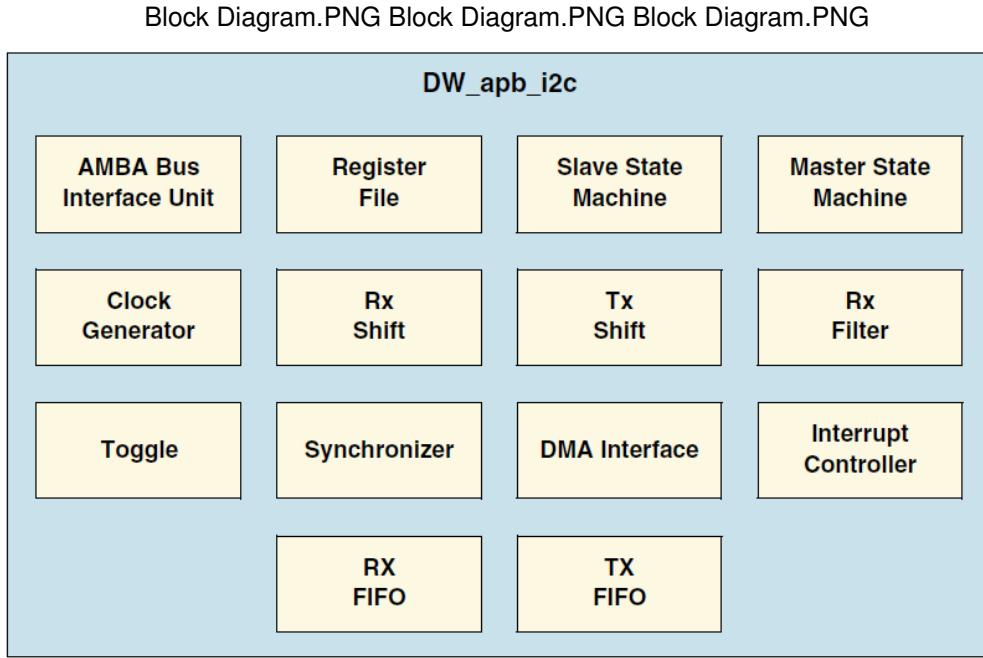
1. Standard mode devices are not upward compatible and should not be incorporated in a fast-mode I2C bus system as they cannot follow the higher transfer rate and unpredictable states would occur.
2. Ultra-Fast mode devices are not downward compatible and should not be incorporated in traditional I2C speeds (High speed, Fast/Fast Mode Plus speed, Standard mode speed) as Ultra-Fast mode follows the higher transfer rate (up to 5Mb/s) with only write transfers and there is no acknowledgment from the slave. An example of high-speed mode devices are LCD displays, high-bit count ADCs, and high capacity EEPROMs. These devices typically need to transfer large amounts of data. Most maintenance and control applications, the common use for the I<sup>A</sup>S<sub>C</sub> bus, typically operate at 100 kHz (in standard and fast modes).

An example of Ultra-Fast speed mode devices are LED controllers and other devices that do not need feedback. These devices typically need to transfer large amounts of data greater than 1Mhz. Any DW\_apb\_i2c device can be attached to an I<sup>A</sup>S<sub>C</sub>-bus and every device can talk with any master, passing information back and forth. There needs to be at least one master (such as a microcontroller or DSP) on the bus but there can be multiple masters, which require them to arbitrate for ownership. Multiple masters and arbitration are explained later in this chapter. The DW\_apb\_i2c also supports SMBus and PMBus protocols for System Management and Power management.

#### Note

In this databook, any reference to SMBus implicitly refers to PMBus also and vice versa.

The DW\_apb\_i2c is made up of an AMBA APB slave interface, an I2C interface, and FIFO logic to maintain coherency between the two interfaces. A simplified block diagram of the component is illustrated in Figure 15.3.



**Figure 15.3:** DW\_apb\_i2c Block Diagram

The following define the file names and functions of the blocks in Figure 15.3:

- AMBA Bus Interface Unit-DW\_apb\_i2c\_biu.v-Takes the APB interface signals and translates them into a common generic interface that allows the register file to be bus protocol-agnostic.
- Register File-DW\_apb\_i2c\_Regfile-Contains configuration registers and is the interface with software.
- Slave State Machine-DW\_apb\_i2c\_slv fsm-Follows the protocol for a slave and monitors bus for address match.
- Master State Machine-DW\_apb\_i2c\_mst fsm-Generates the I2C protocol for the master transfers.
- Clock Generator-DW\_apb\_i2c\_clk\_gen.v-Calculates the required timing to do the following:
  - Generate the SCL clock when configured as a master
  - Check for bus idle
  - Generate a START and a STOP
  - Setup the data and hold the data
- Rx Shift-DW\_apb\_i2c\_rx\_shift-Takes data into the design and extracts it in byte format.
- Tx Shift-DW\_apb\_i2c\_tx\_shift-Presents data supplied by CPU for transfer on the I2C bus.
- Rx Filter-DW\_apb\_i2c\_rx\_filter-Detects the events in the bus; for example, start, stop and arbitration lost.
- Toggle-DW\_apb\_i2c\_toggle-Generates pulses on both sides and toggles to transfer signals across clock domains.
- Synchronizer-DW\_apb\_i2c\_sync-Transfers signals from one clock domain to another.
- DMA Interface-DW\_apb\_i2c\_dma-Generates the handshaking signals to the central DMA controller in order to automate the data transfer without CPU intervention.

- Interrupt Controller-DW\_apb\_i2c\_intctl-Generates the raw interrupt and interrupt flags, allowing them to be set and cleared.
- RX FIFO/TX FIFO-DW\_apb\_i2c\_fifo-Holds the RX FIFO and TX FIFO register banks and controllers, along with their status levels.

**Note**

If PCLK and IC\_CLK are asynchronous (IC\_CLK\_TYPE=ASYNC) then the following condition must be met for DW\_apb\_i2c to function properly:

- When IC\_HAS\_ASYNC\_FIFO = 0,  $(SCL\_LOW\_COUNT * ic\_clk\_period) > (3 * pclk\_period + 3 * ic\_clk\_period))$  Where, SCL\_LOW\_COUNT Specifies the low count value in terms of ic\_clk for the respective speed mode.
- When IC\_HAS\_ASYNC\_FIFO = 1,  $pclk\_period < (9 * scl\_period)/2$  Where, pclk\_period Specifies the clock period of the application clock. scl\_period Specifies the SCL period.

## 15.2.2 I2C Terminology

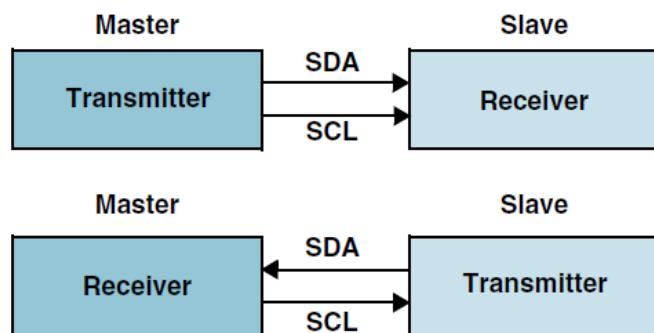
The following terms are used throughout this manual and are defined as follows:

### 15.2.2.1 I2C Bus Terms

The following terms relate to how the role of the I2C device and how it interacts with other I2C devices on the bus.

- Transmitter - the device that sends data to the bus. A transmitter can either be a device that initiates the data transmission to the bus (a master-transmitter) or responds to a request from the master to send data to the bus (a slave-transmitter).
- Receiver - the device that receives data from the bus. A receiver can either be a device that receives data on its own request (a master-receiver) or in response to a request from the master (a slave-receiver).
- Master – the component that initializes a transfer (START command), generates the clock (SCL) signal and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
- Slave - the device addressed by the master. A slave can be either receiver or transmitter.

These concepts are illustrated in Figure 15.4.



**Figure 15.4: Master/Slave and Transmitter/Receiver Relationships**

- Multi-master - the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- Arbitration - the predefined procedure that authorizes only one master at a time to take control of the bus. For more information about this behavior, refer to "Multiple Master Arbitration".
- Synchronization - the predefined procedure that synchronizes the clock signals provided by two or more masters. For more information about this feature, refer to "Clock Synchronization".
- SDA - data signal line (Serial DAta)
- SCL - clock signal line (Serial CLock)

### 15.2.2.2 Bus Transfer Terms

The following terms are specific to data transfers that occur to/from the I2C bus.

- START (RESTART) - data transfer begins with a START or RESTART condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes busy.

Note

START and RESTART conditions are functionally identical.

- STOP - data transfer is terminated by a STOP condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle once again. The bus stays busy if a RESTART is generated instead of a STOP condition.

### 15.2.3 I2C Behavior

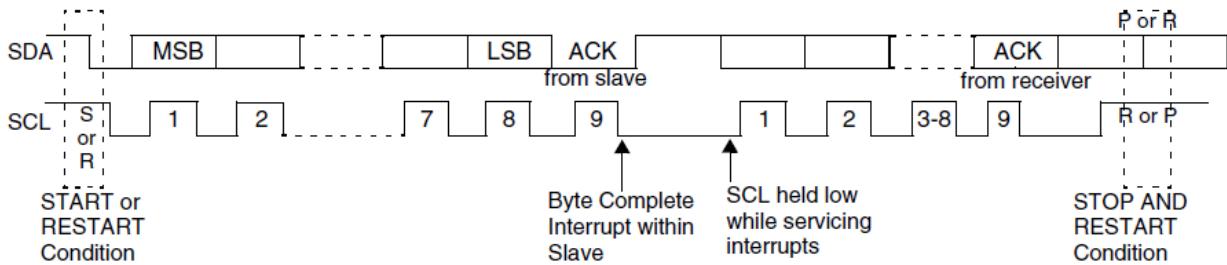
The DW\_apb\_i2c can be controlled via software to be either:

- An I2C master only, communicating with other I2C slaves; OR
- An I2C slave only, communicating with one more I2C masters.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership. Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address. If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in Figure 15.5. In Ultra-Fast Speed Mode, the master can issue only the write

transfers to the slaves with always not acknowledging (NACK) from the slaves. Read transfers are not allowed in this mode.

transfer on the I2C Bus.PNG transfer on the I2C Bus.PNG transfer on the I2C Bus.PNG



**Figure 15.5:** Data transfer on the I2C Bus

The DW\_apb\_i2c is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages. The I2C protocols implemented in DW\_apb\_i2c are described in more details in "I2C Protocols".

#### 15.2.3.1 START and STOP Generation

When operating as an I2C master, putting data into the transmit FIFO causes the DW\_apb\_i2c to generate a START condition on the I2C bus. If the IC\_EMPTYFIFO\_HOLD\_MASTER\_EN parameter is set to 0, allowing the transmit FIFO to empty causes the DW\_apb\_i2c to generate a STOP condition on the I2C bus. If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN is set to 1, then writing a 1 to IC\_DATA\_CMD[9] causes the DW\_apb\_i2c to generate a STOP condition on the I2C bus; a STOP condition is not issued if this bit is not set, even if the transmit FIFO is empty. When operating as a slave, the DW\_apb\_i2c does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the DW\_apb\_i2c, it holds the SCL line low until read data has been supplied to it. This stalls the I2C bus until read data is provided to the slave DW\_apb\_i2c, or the DW\_apb\_i2c slave is disabled by writing a 0 to bit 0 of the IC\_ENABLE register.

#### 15.2.3.2 Combined Formats

The DW\_apb\_i2c supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes. The DW\_apb\_i2c does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions. To initiate combined format transfers, IC\_CON.IC\_RESTART\_EN should be set to 1. With this value set and operating as a master, when the DW\_apb\_i2c completes an I2C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I2C transfer completes—depending on the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN:

- Either a STOP is issued or,
- IC\_DATA\_CMD[9] is checked and:
  - If set to 1, a STOP bit is issued.

- If set to 0, the SCL is held low until the next command is written to the transmit FIFO.

For more details, refer to "Tx FIFO Management and START, STOP and RESTART Generation".

#### Note

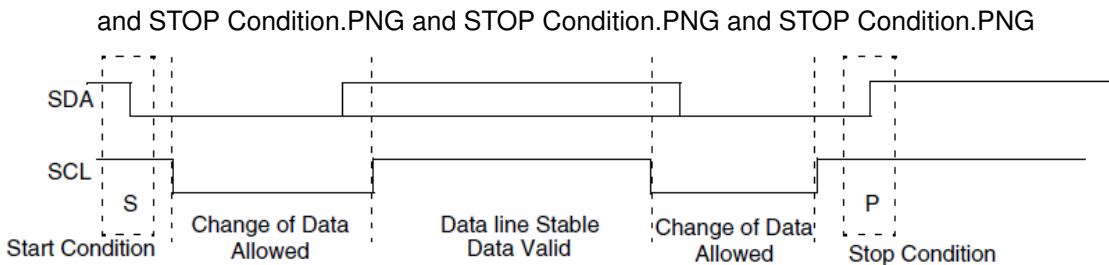
Mixed write and read transactions in both 7-bit and 10-bit addressing modes are not applicable for Ultra-Fast Mode (IC\_ULTRA\_FAST\_MODE=1) as read transfers are not supported in Ultra-Fast Mode.

## 15.2.4 I2C Protocols

The DW\_apb\_i2c has the protocols discussed in this section.

### 15.2.4.1 START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. Figure 15.6 shows the timing of the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.



**Figure 15.6:** START and STOP Condition

#### Note

The signal transitions for the START/STOP conditions, as depicted in Figure 15.6, reflect those observed at the output signals of the Master driving the I2C bus. Care should be taken when observing the SDA/SCL signals at the input signals of the Slave(s), because unequal line delays may result in an incorrect SDA/SCL timing relationship.

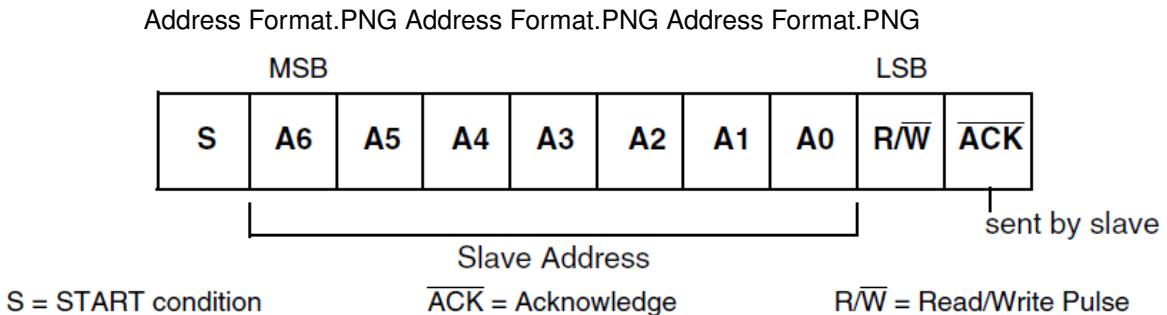
### 15.2.4.2 Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

#### 15.2.4.2.1 7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit

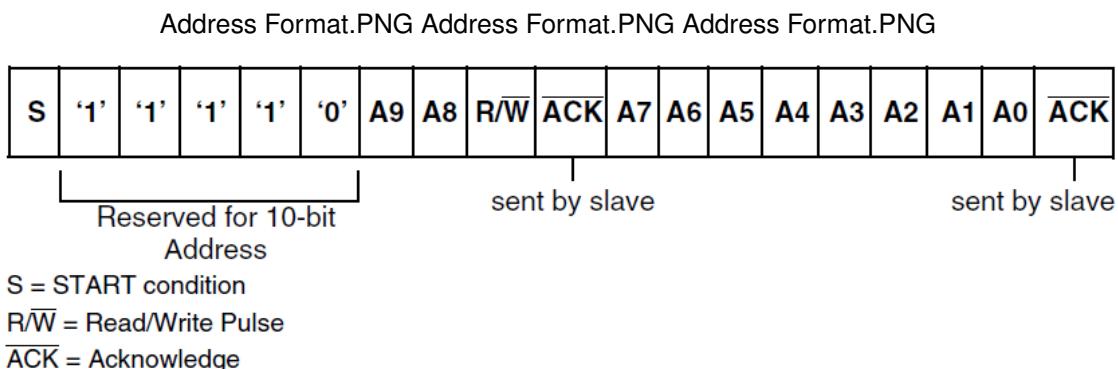
(bit 0) is the R/W bit as shown in Figure 15.7. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.



**Figure 15.7:** 7-bit Address Format

#### 15.2.4.2.2 10-bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 15.8 shows the 10-bit address format.



**Figure 15.8:** 10-bit Address Format

Table 15.1 defines the special purpose and reserved first byte addresses.

<b>Slave Address</b>	<b>R/W Bit</b>	<b>Description</b>
0000 000	0	General Call Address. DW_apb_i2c places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START byte. For more details, refer to "START BYTE Transfer Protocol".
0000 001	X	CBUS address. DW_apb_i2c ignores these accesses.
0000 010	X	Reserved.
0000 011	X	Reserved.

**Table 15.1:** I<sup>2</sup>C/SMBus Definition of Bits in First Byte (Continued on next page)

continued from previous page		
0000 1XX	X	High-speed master code (for more information, refer to "Multiple Master Arbitration").
1111 1XX	X	Reserved.
1111 0XX	X	10-bit slave addressing.
0001 000	X	SMbus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

**Table 15.1:** I2C/SMBus Definition of Bits in First Byte

DW\_apb\_i2c does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I2C components.

#### 15.2.4.3 Transmitting and Receiving Protocol

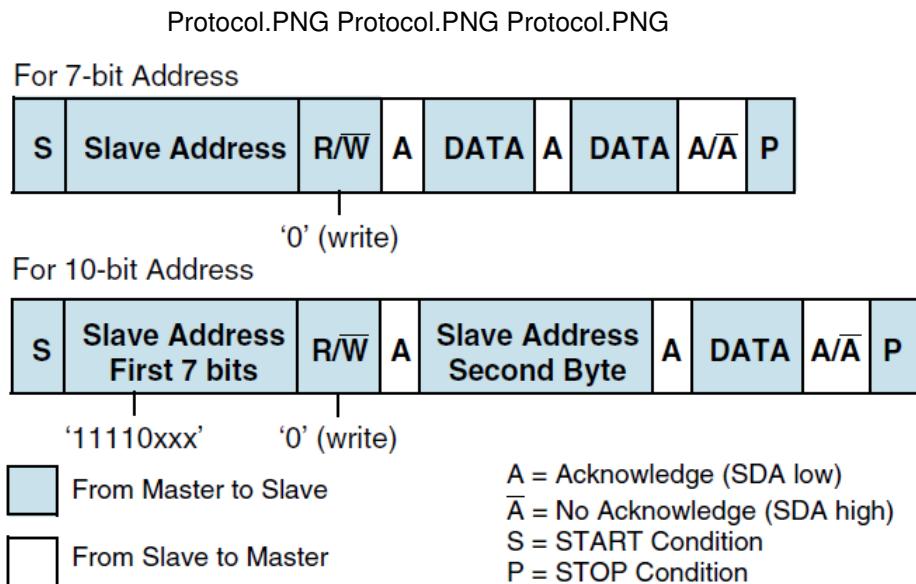
The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively.

##### 15.2.4.3.1 Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer. If the master-transmitter is transmitting data as shown in Figure 15.9, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

##### Note

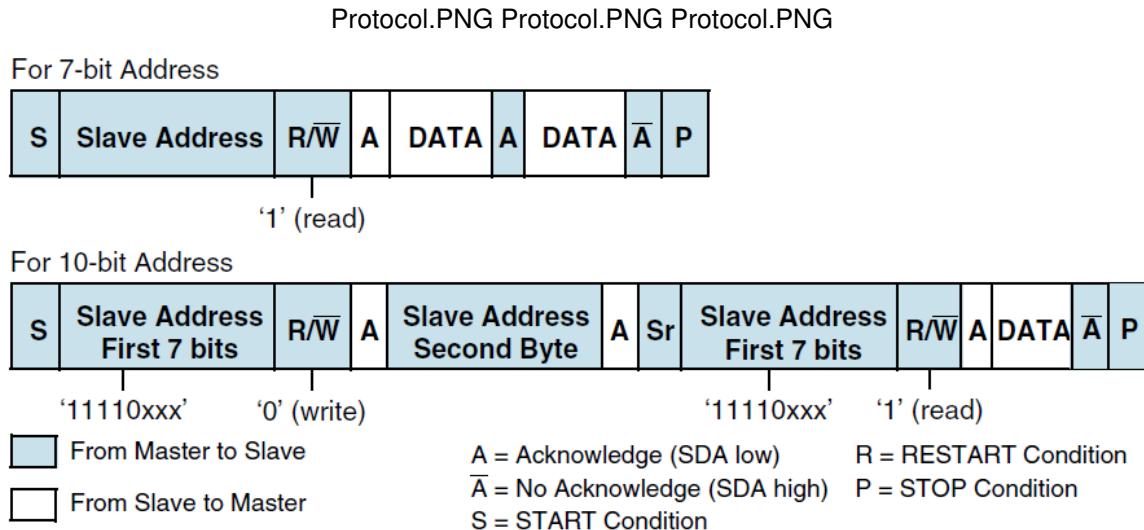
In Ultra-Fast Mode, the slave-receiver always responds with the No Acknowledge signal (NACK) for the Address and the write data from the Master.



**Figure 15.9:** Master-Transmitter Protocol

#### **15.2.4.3.2 Master-Receiver and Slave-Transmitter**

If the master is receiving data as shown in Figure 15.10, then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition.



**Figure 15.10:** Master-Receiver Protocol

When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in master mode,

the DW\_apb\_i2c can then communicate with the same slave using a transfer of a different direction. For a description of the combined format transactions that the DW\_apb\_i2c supports, refer to "Combined Formats".

#### Note

The DW\_apb\_i2c must be completely disabled-if I2C\_DYNAMIC\_TAR\_UPDATE = 0-or inactive on the serial port-if I2C\_DYNAMIC\_TAR\_UPDATE = 1-before the target slave address register (IC\_TAR) can be reprogrammed.

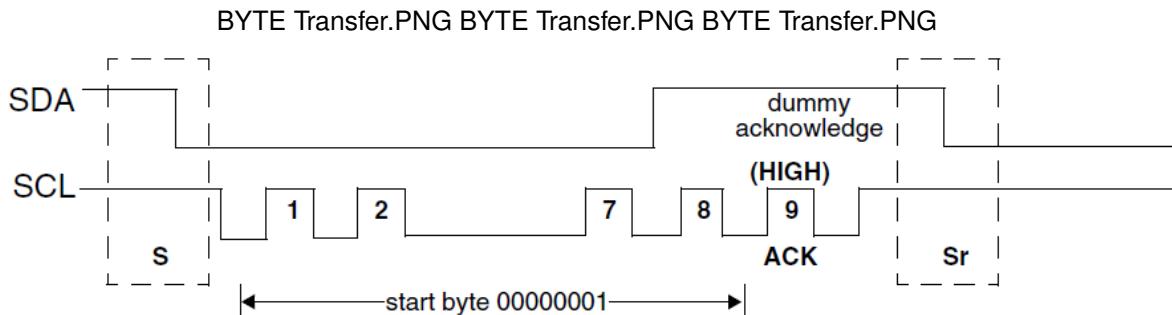
#### Note

In Ultra-Fast mode, the Master receiver and Slave Transmitter is not applicable, as read transfers are not supported.

#### 15.2.4.4 START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I2C hardware module. When the DW\_apb\_i2c is addressed as a slave, it always samples the I2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when DW\_apb\_i2c is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it.

This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Figure 15.11. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master.



**Figure 15.11:** START BYTE Transfer

The START BYTE procedure is as follows:

1. Master generates a START condition.
2. Master transmits the START byte (0000 0001).
3. Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
4. No slave sets the ACK signal to 0.
5. Master generates a RESTART (R) condition.

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated.

### 15.2.5 Tx FIFO Management and START, STOP and RESTART Generation

When operating as a master, the DW\_apb\_i2c component supports two modes of Tx FIFO management. You use the IC\_EMPTYFIFO\_HOLD\_MASTER\_EN parameter to select between these two modes:

- IC\_EMPTYFIFO\_HOLD\_MASTER\_EN equals 0, illustrated in Figure 15.12
- IC\_EMPTYFIFO\_HOLD\_MASTER\_EN equals 1, illustrated in Figure 15.15

#### 15.2.5.1 Tx FIFO Management When IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0

When the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN is 0, the component generates a STOP on the bus whenever the Tx FIFO becomes empty. If RESTART generation capability is enabled, the component generates a RESTART when the direction of the transfer in the Tx FIFO commands changes from Read to Write or vice-versa; if RESTART is not enabled, a STOP followed by a START is generated in this situation. Figure 15.12 shows the bits in the IC\_DATA\_CMD register if IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0.

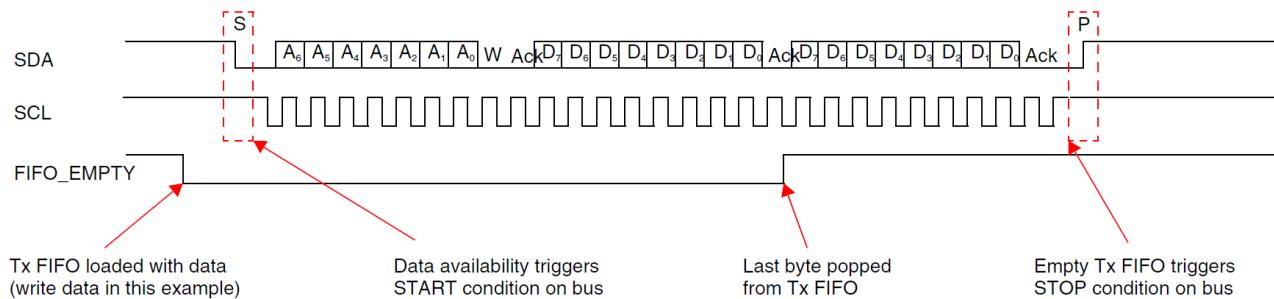
IC_DATA_CMD	CMD	DATA	
	8      7		0

DATA –Read/Write field; data retrieved from slave is read from this field; data to be sent to slave is written to this field.

CMD –Write-only field; this bit determines whether transfer to be carried out is Read (CMD=1) or Write (CMD=0)

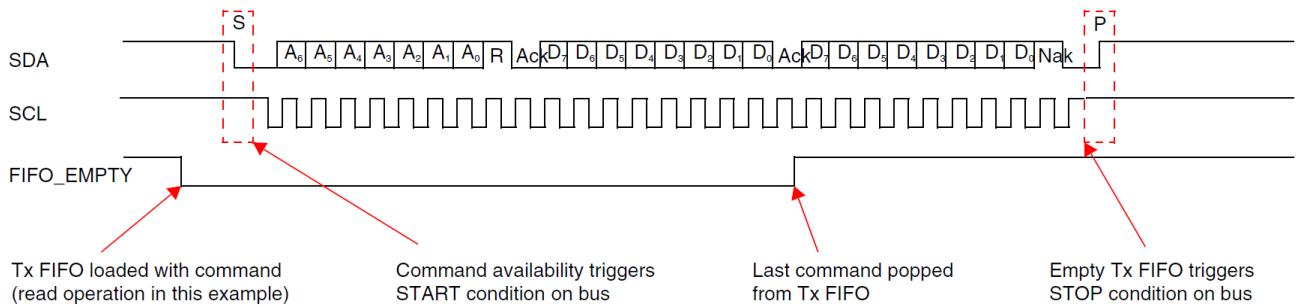
**Figure 15.12:** IC\_DATA\_CMD Register if IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0

Figure 15.13 shows a timing diagram that illustrates the behavior of the DW\_apb\_i2c when Tx FIFO becomes empty while operating as a master transmitter when IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=0.



**Figure 15.13:** Master Transmitter - Tx FIFO Becomes Empty If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0

Figure 15.14 shows a timing diagram that illustrates the behavior of the DW\_apb\_i2c when Tx FIFO becomes empty while operating as a master receiver when IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=0.



**Figure 15.14:** Master Receiver - Tx FIFO Becomes Empty If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0

#### Note

When IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0, if the TX\_EMPTY\_CTRL bit of the IC\_CON register is set to 1 and the transmit threshold (TX\_THLD) is set to 0, then one I2C frame (consisting of multiple commands) always breaks into multiple frames based on the number of commands in each frame. This is because DW\_apb\_i2c issues a TX\_EMPTY interrupt after the end of each data (TX\_EMPTY\_CTRL=1) is sent on the line and the data (TX\_THLD=0) is pushed based on TX\_EMPTY interrupt.

### 15.2.6 Tx FIFO Management When IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1

When the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN is 1, the component does not generate a STOP if the Tx FIFO becomes empty; in this situation the component holds the SCL line low, stalling the bus until a new entry is available in the Tx FIFO. A STOP condition is generated only when the user specifically requests it by setting bit 9 (Stop bit) of the command written to IC\_DATA\_CMD register. Figure 15.15 shows the bits in the IC\_DATA\_CMD register if IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1.

IC_DATA_CMD	Restart	Stop	CMD	DATA	
	10	9	8	7	0

DATA –Read/Write field; data retrieved from slave is read from this field; data to be sent to slave is written to this field

CMD –Write-only field; this bit determines whether transfer to be carried out is Read (CMD=1) or Write (CMD=0)

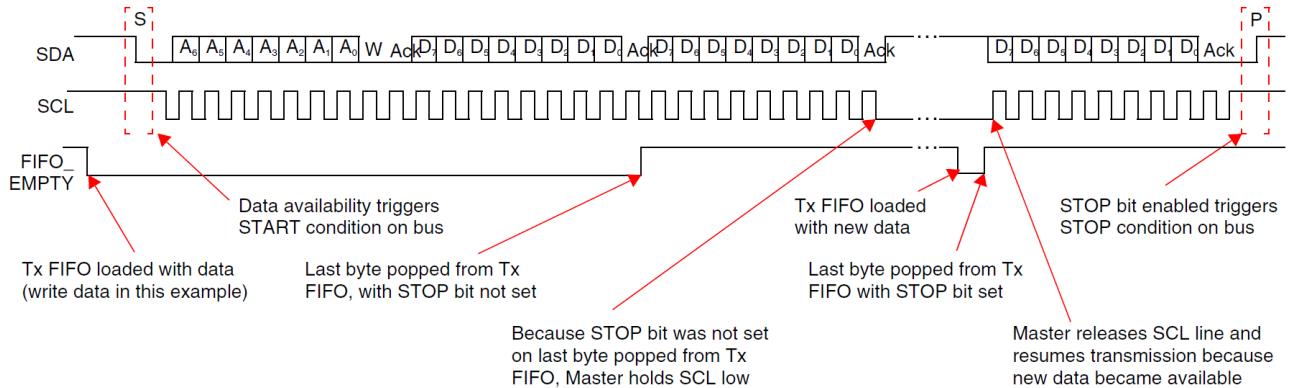
Stop –Write-only field; this bit determines whether STOP is generated after data byte is sent or received

Restart – Write-only field; this bit determines whether RESTART (or STOP followed by START in case of restart capability is not enabled) is generated before data byte is sent or received

**Figure 15.15:** IC\_DATA\_CMD Register if IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1

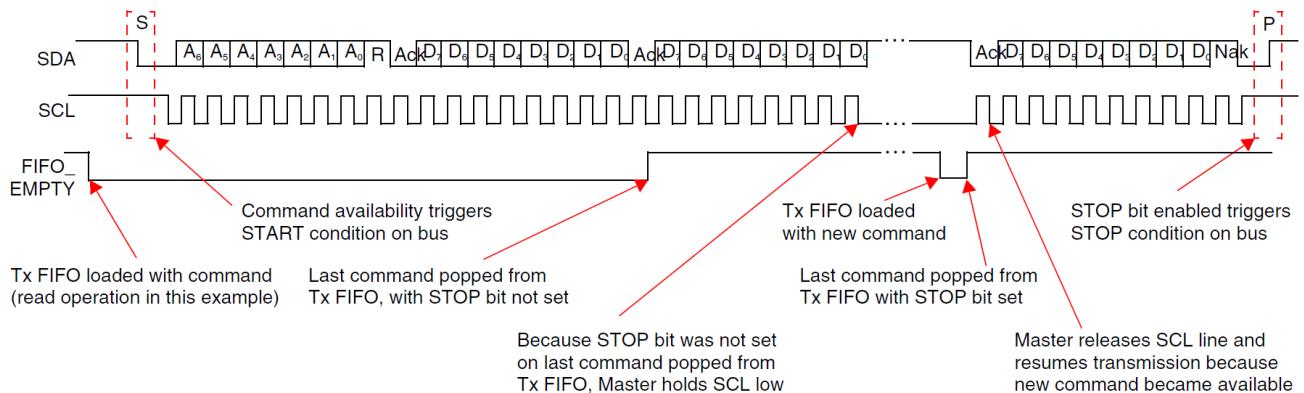
Figure 15.16 illustrates the behavior of the DW\_apb\_i2c when the Tx FIFO becomes empty while operating as a

master transmitter, as well as showing the generation of a STOP condition when IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1.



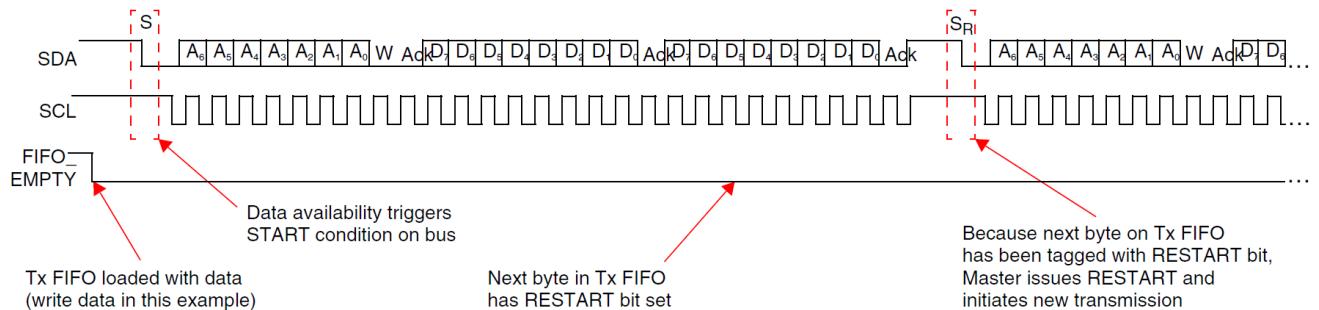
**Figure 15.16:** Master Transmitter - Tx FIFO Empties/STOP Generation If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1

Figure 15.17 illustrates the behavior of the DW\_apb\_i2c when the Tx FIFO becomes empty while operating as a master receiver, as well as showing the generation of a STOP condition when IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1.



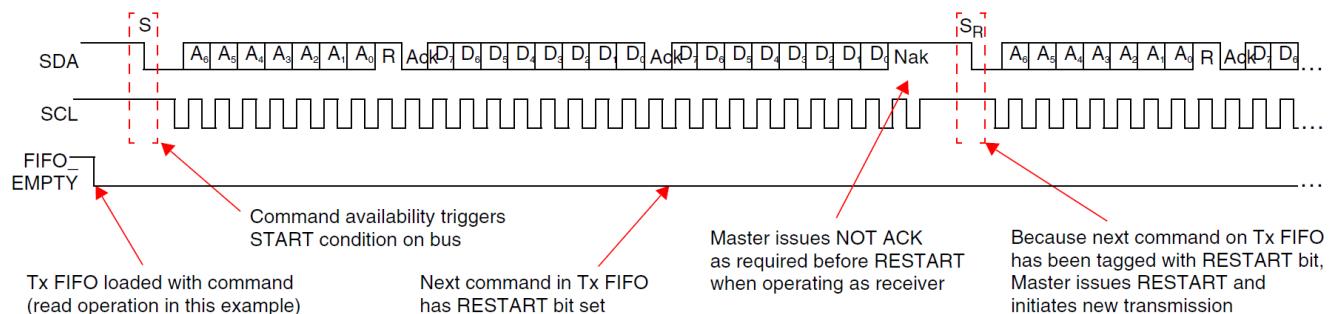
**Figure 15.17:** Master Receiver - Tx FIFO Empties/STOP Generation If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1

Figure 15.18 and Figure 15.19 illustrate configurations where the user can control the generation of RESTART conditions on the I2C bus. If bit 10 (Restart) of the IC\_DATA\_CMD register is set and the restart capability is enabled (IC\_RESTART\_EN=1), a RESTART is generated before the data byte is written to or read from the slave. If the restart capability is not enabled a STOP followed by a START is generated in place of the RESTART. Figure 15.18 illustrates this situation during operation as a master transmitter.



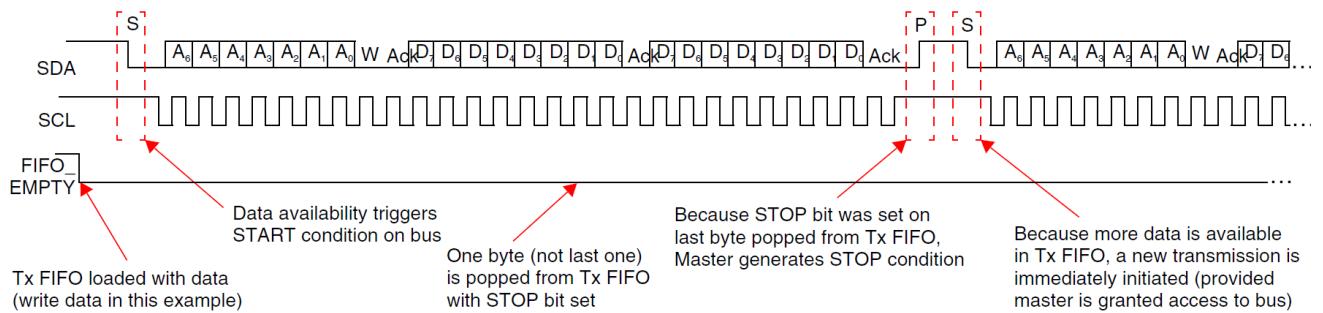
**Figure 15.18:** Master Transmitter - Restart Bit of IC\_DATA\_CMD Is Set (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1)

Figure 15.19 illustrates the same situation, but during operation as a master receiver.



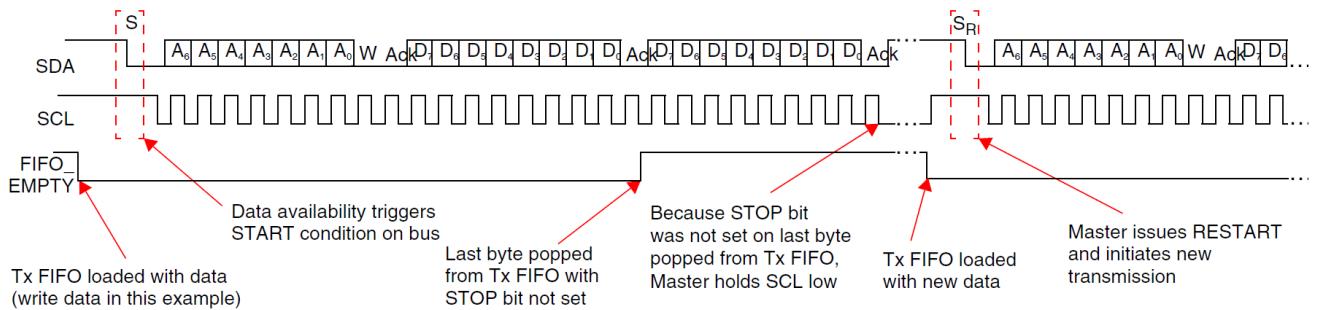
**Figure 15.19:** Master Receiver - Restart Bit of IC\_DATA\_CMD Is Set (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1)

Figure 15.20 illustrates operation as a master transmitter where the Stop bit of the IC\_DATA\_CMD register is set and the Tx FIFO is not empty (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1).



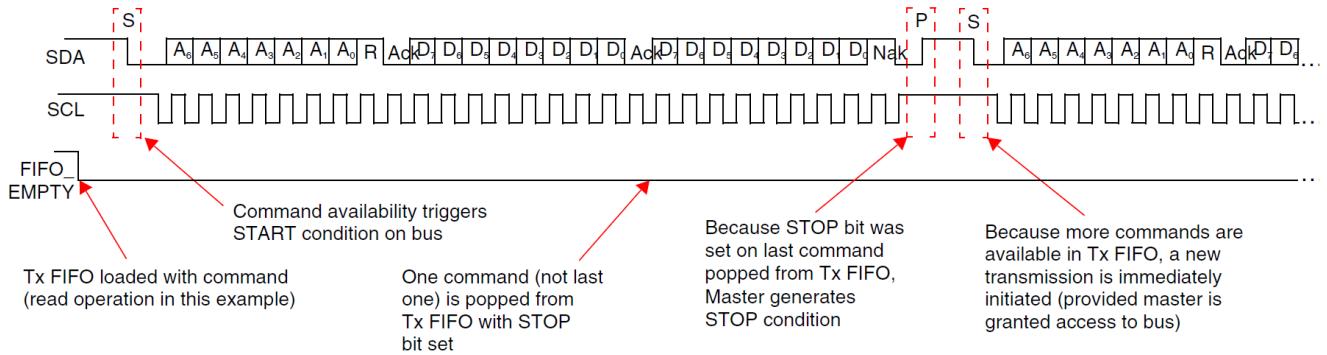
**Figure 15.20:** Master Transmitter - Stop Bit of IC\_DATA\_CMD Set/Tx FIFO Not Empty

Figure 15.21 illustrates operation as a master transmitter where the first byte loaded into the Tx FIFO is allowed to go empty with the Restart bit set (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1).



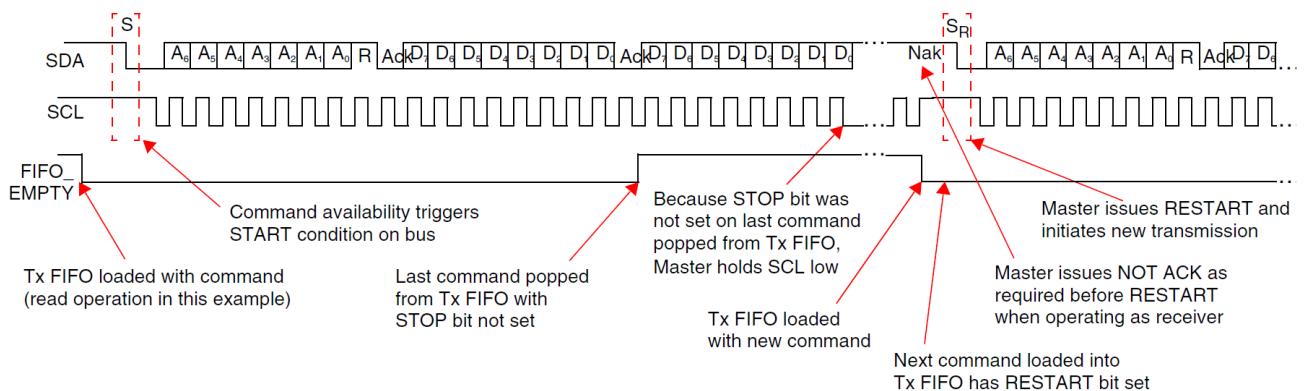
**Figure 15.21:** Master Transmitter - First Byte Loaded Into Tx FIFO Allowed to Empty, Restart Bit Set (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1)

Figure 15.22 illustrates operation as a master receiver where the Stop bit of the IC\_DATA\_CMD register is set and the Tx FIFO is not empty (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1).



**Figure 15.22:** Master Receiver - Stop Bit of IC\_DATA\_CMD Set/Tx FIFO Not Empty (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1 and IC\_ULTRA\_FAST\_MODE=0)

Figure 15.23 illustrates operation as a master receiver where the first command loaded after the Tx FIFO is allowed to empty and the Restart bit is set (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1).

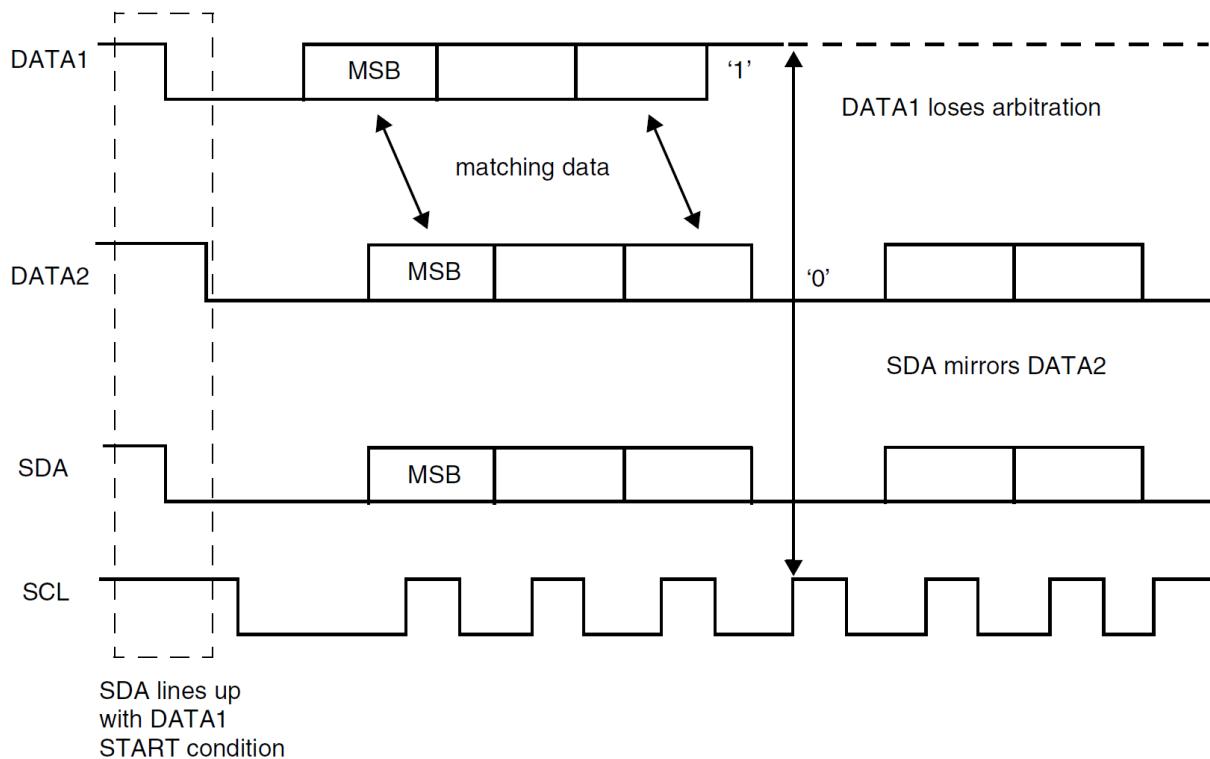


**Figure 15.23:** Master Receiver - First Command Loaded After Tx FIFO Allowed to Empty/Restart Bit Set (IC\_EMPTYFIFO\_HOLD\_MASTER\_EN=1 and IC\_ULTRA\_FAST\_MODE=0)

### 15.2.7 Multiple Master Arbitration

The DW\_apb\_i2c bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I<sup>2</sup>C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a microcontroller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state. Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, the DW\_apb\_i2c will stop generating SCL (ic\_clk\_oe). Figure 15.24 illustrates the timing of when two masters are arbitrating on the bus.



**Figure 15.24:** Multiple Master Arbitration

For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bitcode is defined by the system designer and is set by writing to the High Speed Master Mode Code Address Register, IC\_HS\_MADDR. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code. Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition

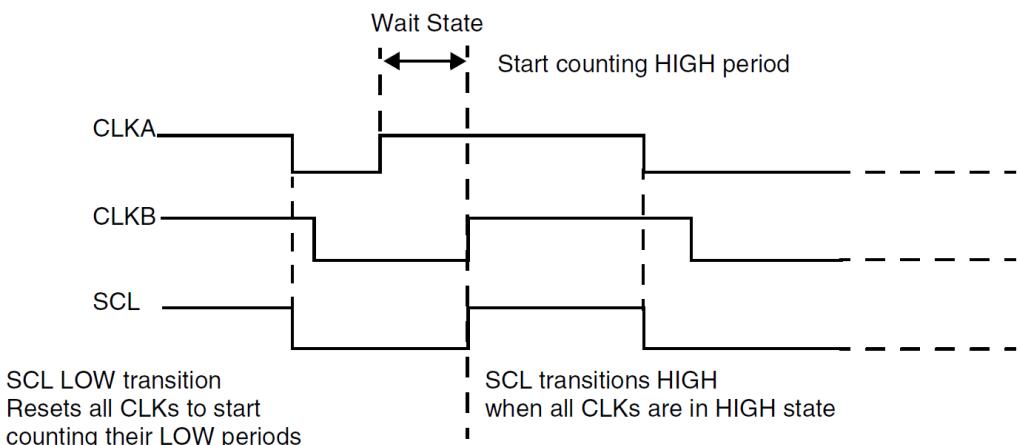
Slaves are not involved in the arbitration process.

**Note**

Multi-master arbitration is not applicable in Ultra-Fast Mode (IC\_ULTRA\_FAST\_MODE=1) as single Master is present.

### 15.2.8 Clock Synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a HIGH wait state until the SCL clock line transitions to 1. All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 0. The masters then counts out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in Figure 15.25. Optionally, slaves may hold the SCL line low to slow down the timing on the I2C bus.



**Figure 15.25:** Multi-Master Clock Synchronization

**Note**

Clock Synchronization is not supported in Ultra-Fast Mode (IC\_ULTRA\_FAST\_MODE=1) as single master is present in the Ultra-Fast Mode system.

## 15.2.9 Operation Modes

This section provides information on operation modes.

### Note

It is important to Note that the DW\_apb\_i2c should only be set to operate as an I2C Master, or I2C Slave, but not both simultaneously. This is achieved by ensuring that bit 6 (IC\_SLAVE\_DISABLE) and 0 (IC\_MASTER\_MODE) of the IC\_CON register are never set to 0 and 1, respectively.

### 15.2.9.1 Slave Mode Operation

This section discusses slave mode procedures.

#### 15.2.9.1.1 Initial Configuration

To use the DW\_apb\_i2c as a slave, perform the following steps:

1. Disable the DW\_apb\_i2c by writing a '0' to bit 0 of the IC\_ENABLE register.
2. Write to the IC\_SAR register (bits 9:0) to set the slave address. This is the address to which the DW\_apb\_i2c responds.
3. Write to the IC\_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the DW\_apb\_i2c in slave-only mode by writing a '0' into bit 6 (IC\_SLAVE\_DISABLE) and a '0' to bit 0 (MASTER\_MODE).

### Note

Slaves and masters do not have to be programmed with the same type of addressing 7- or 10- bit address. For instance, a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

4. Enable the DW\_apb\_i2c by writing a '1' in bit 0 of the IC\_ENABLE register.

### Note

Depending on the reset values chosen, steps 2 and 3 may not be necessary because the reset values can be configured. For instance, if the device is only going to be a master, there would be no need to set the slave address because you can configure DW\_apb\_i2c to have the slave disabled after reset and to enable the master after reset. The values stored are static and do not need to be reprogrammed if the DW\_apb\_i2c is disabled.

### Attention

It is recommended that the DW\_apb\_i2c Slave be brought out of reset only when the I2C bus is IDLE. De-asserting the reset when a transfer is ongoing on the bus causes internal synchronization flip-flops used to synchronize SDA and SCL to toggle from a reset value of 1 to the actual value on the bus. This can result in SDA toggling from 1 to 0 while SCL is 1, thereby causing a false START condition to be detected by the DW\_apb\_i2c Slave. This scenario can also be avoided by configuring the DW\_apb\_i2c with IC\_SLAVE\_DISABLE = 1 and IC\_MASTER\_MODE = 1 so that the Slave interface is disabled after reset. It can then be enabled by programming IC\_CON[0] = 0 and IC\_CON[6] = 0 after the internal SDA and SCL have synchronized to the value on the bus; this takes approximately 6 ic\_clk cycles after reset de-assertion.

### 15.2.9.1.2 Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses the DW\_apb\_i2c and requests data, the DW\_apb\_i2c acts as a slave-transmitter and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the slave address in the IC\_SAR register of the DW\_apb\_i2c.
2. The DW\_apb\_i2c acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
3. The DW\_apb\_i2c asserts the RD\_REQ interrupt (bit 5 of the IC\_RAW\_INTR\_STAT register) and holds the SCL line low. It is in a wait state until software responds. If the RD\_REQ interrupt has been masked, due to IC\_INTR\_MASK[5] register (M\_RD\_REQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the IC\_RAW\_INTR\_STAT register.
  - (a) Reads that indicate IC\_RAW\_INTR\_STAT[5] (R\_RD\_REQ bit field) being set to 1 must be treated as the equivalent of the RD\_REQ interrupt being asserted.
  - (b) Software must then act to satisfy the I2C transfer.
  - (c) The timing interval used should be in the order of 10 times the fastest SCL clock period the DW\_apb\_i2c can handle. For example, for 400 kb/s, the timing interval is 25us.

Note

The value of 10 is recommended here because this is approximately the amount of time required for a single byte of data transferred on the I2C bus.

4. If there is any data remaining in the Tx FIFO before receiving the read request, then the DW\_apb\_i2c asserts a TX\_ABRT interrupt (bit 6 of the IC\_RAW\_INTR\_STAT register) to flush the old data from the TX FIFO.

Note

Because the DW\_apb\_i2c's Tx FIFO is forced into a flushed/reset state whenever a TX\_ABRT event occurs, it is necessary for software to release the DW\_apb\_i2c from this state by reading the IC\_CLR\_TX\_ABRT register before attempting to write into the Tx FIFO. See register IC\_RAW\_INTR\_STAT for more details.

If the TX\_ABRT interrupt has been masked, due to of IC\_INTR\_MASK[6] register (M\_TX\_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to read the IC\_RAW\_INTR\_STAT register.

- (a) Reads that indicate bit 6 (R\_TX\_ABRT) being set to 1 must be treated as the equivalent of the TX\_ABRT interrupt being asserted.
- (b) There is no further action required from software.
- (c) The timing interval used should be similar to that described in the previous step for the IC\_RAW\_INTR\_STAT[5] register.
5. Software writes to the IC\_DATA\_CMD register with the data to be written (by writing a '0' in bit 8).
6. Software must clear the RD\_REQ and TX\_ABRT interrupts (bits 5 and 6, respectively) of the IC\_RAW\_INTR\_STAT register before proceeding. If the RD\_REQ and/or TX\_ABRT interrupts have been masked, then clearing of the IC\_RAW\_INTR\_STAT register will have already been performed when either the R\_RD\_REQ or R\_TX\_ABRT bit has been read as 1.
7. The DW\_apb\_i2c releases the SCL and transmits the byte.
8. The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

**Note**

Slave-Transmitter Operation for a Single Byte is not applicable in Ultra-Fast Mode as Read transfers are not supported.

#### 15.2.9.1.3 Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses the DW\_apb\_i2c and is sending data, the DW\_apb\_i2c acts as a slave-receiver and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the DW\_apb\_i2c's slave address in the IC\_SAR register.
2. The DW\_apb\_i2c acknowledges the sent address and recognizes the direction of the transfer to indicate that the DW\_apb\_i2c is acting as a slave-receiver.
3. DW\_apb\_i2c receives the transmitted byte and places it in the receive buffer.

**Note**

If the Rx FIFO is completely filled with data when a byte is pushed, and IC\_RX\_FULL\_HLD\_BUS\_EN = 0, then an overflow occurs and the DW\_apb\_i2c continues with subsequent I2C transfers. Because a NACK is not generated, software must recognize the overflow when indicated by the DW\_apb\_i2c (by the R\_RX\_OVER bit in the IC\_INTR\_STAT register) and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the Rx FIFO before the latter overflows, as there is no way to re-apply pressure to the remote transmitting master. You must select a deep enough Rx FIFO depth to satisfy the interrupt service interval of the system. If the Rx FIFO is completely filled with data when a byte is pushed, and IC\_RX\_FULL\_HLD\_BUS\_EN = 1, then the DW\_apb\_i2c slave holds the I2C SCL line low until the Rx FIFO has some space, and then continues with the next read request.

4. DW\_apb\_i2c asserts the RX\_FULL interrupt (IC\_RAW\_INTR\_STAT[2] register). If the RX\_FULL interrupt has been masked, due to setting IC\_INTR\_MASK[2] register to 0 or setting IC\_TX\_TL to a value larger than 0, then it is recommended that a timing routine (described in "Slave-Transmitter Operation for a Single Byte") be implemented for periodic reads of the IC\_STATUS register. Reads of the IC\_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX\_FULL interrupt being asserted.
5. Software may read the byte from the IC\_DATA\_CMD register (bits 7:0).
6. The other master device may hold the I2C bus by issuing a RESTART condition, or release the bus by issuing a STOP condition.

#### 15.2.9.1.4 Slave-Transfer Operation For Bulk Transfers

In the standard I2C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TX FIFO. When a slave (slave-transmitter) is issued with a read request (RD\_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TX FIFO. DW\_apb\_i2c is designed to handle more data in the TX FIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TX FIFO. This mode only occurs when DW\_apb\_i2c is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the DW\_apb\_i2c holds the I2C SCL line low while it raises the read request interrupt (RD\_REQ) and waits for data to be written into the TX FIFO before it can be sent to the remote master. If the RD\_REQ interrupt is masked, due to bit 5 (M\_RD\_REQ) of the IC\_INTR\_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the IC\_RAW\_INTR\_STAT register. Reads of IC\_RAW\_INTR\_STAT that return bit 5 (R\_RD\_REQ) set to 1 must be treated as the equivalent of the RD\_REQ

interrupt referred to in this section. This timing routine is similar to that described in "Slave-Transmitter Operation for a Single Byte". The RD\_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the Tx FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte, then the slave must raise the RD\_REQ again because the master is requesting for more data. If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses DW\_apb\_i2c and requests data, the Tx FIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the DW\_apb\_i2c slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the Tx FIFO. There is no need to hold the SCL line low or to issue RD\_REQ again. If the remote master is to receive n bytes from the DW\_apb\_i2c but the programmer wrote a number of bytes larger than n to the Tx FIFO, then when the slave finishes sending the requested n bytes, it clears the Tx FIFO and ignores any excess bytes. The DW\_apb\_i2c generates a transmit abort (TX\_ABRT) event to indicate the clearing of the Tx FIFO in this example. At the time an ACK/NACK is expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data in the Tx FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the Tx FIFO is cleared at that time.

**Note**

Slave Transmitter Operation for Bulk Transfers is not applicable in Ultra-Fast Mode (IC\_ULTRA\_FAST\_MODE=1) as Master Read Transfers are not supported.

### 15.2.9.2 Master Mode Operation

This section discusses master mode procedures.

#### 15.2.9.2.1 Initial Configuration

The initial configuration procedure for Master Mode Operation depends on the configuration parameter I2C\_DYNAMIC\_TAR\_UP. When set to "Yes" (1), the target address and address format can be changed dynamically without having to disable DW\_apb\_i2c. This parameter only applies to when DW\_apb\_i2c is acting as a master because the slave requires the component to be disabled before any changes can be made to the address. For more information about this parameter, see "Parameter Descriptions". For more information about how this parameter affects the IC\_TAR register, see "Register Descriptions". The procedures are very similar and are only different with regard to where the IC\_10BITADDR\_MASTER bit is set (either bit 4 of IC\_CON register or bit 12 of IC\_TAR register).

##### 15.2.9.2.1.1 I2C\_DYNAMIC\_TAR\_UPDATE = 0

To use the DW\_apb\_i2c as a master when the I2C\_DYNAMIC\_TAR\_UPDATE configuration parameter is set to "No" (0), perform the following steps:

1. Disable the DW\_apb\_i2c by writing 0 to bit 0 of the IC\_ENABLE register.
2. Write to the IC\_CON register to set the maximum speed mode supported (bits 2:1) and the desired speed of the DW\_apb\_i2c master-initiated transfers, either 7-bit or 10-bit addressing (bit 4). Ensure that bit 6 (IC\_SLAVE\_DISABLE) is written with a '1' and bit 0 (MASTER\_MODE) is written with a '1'.

**Note**

Slaves and masters do not have to be programmed with the same type of addressing 7- or 10- bit address. For instance, a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

3. Write to the IC\_TAR register the address of the I2C device to be addressed (bits 9:0). This register also indicates whether a General Call or a START BYTE command is going to be performed by I2C.

4. Only applicable for high-speed mode transfers. Write to the IC\_HS\_MADDR register the desired master code for the DW\_apb\_i2c. The master code is programmer-defined.
5. Enable the DW\_apb\_i2c by writing a 1 to bit 0 of the IC\_ENABLE register.
6. Now write transfer direction and data to be sent to the IC\_DATA\_CMD register. If the IC\_DATA\_CMD register is written before the DW\_apb\_i2c is enabled, the data and commands are lost as the buffers are kept cleared when DW\_apb\_i2c is disabled. This step generates the START condition and the address byte on the DW\_apb\_i2c. Once DW\_apb\_i2c is enabled and there is data in the TX FIFO, DW\_apb\_i2c starts reading the data.

Note

Depending on the reset values chosen, steps 2, 3, 4, and 5 may not be necessary because the reset values can be configured. The values stored are static and do not need to be reprogrammed if the DW\_apb\_i2c is disabled, with the exception of the transfer direction and data.

#### 15.2.9.2.1.2 I2C\_DYNAMIC\_TAR\_UPDATE = 1

To use the DW\_apb\_i2c as a master when the I2C\_DYNAMIC\_TAR\_UPDATE configuration parameter is set to "Yes" (1), perform the following steps:

1. Disable the DW\_apb\_i2c by writing 0 to bit 0 of the IC\_ENABLE register.
2. Write to the IC\_CON register to set the maximum speed mode supported for slave operation (bits 2:1) and to specify whether the DW\_apb\_i2c starts its transfers in 7/10 bit addressing mode when the device is a slave (bit 3).
3. Write to the IC\_TAR register the address of the I2C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I2C. The desired speed of the DW\_apb\_i2c master-initiated transfers, either 7-bit or 10-bit addressing, is controlled by the IC\_10BITADDR\_MASTER bit field (bit 12).
4. Only applicable for high-speed mode transfers. Write to the IC\_HS\_MADDR register the desired master code for the DW\_apb\_i2c. The master code is programmer-defined.
5. Enable the DW\_apb\_i2c by writing a 1 to bit 0 of the IC\_ENABLE register.
6. Now write the transfer direction and data to be sent to the IC\_DATA\_CMD register. If the IC\_DATA\_CMD register is written before the DW\_apb\_i2c is enabled, the data and commands are lost as the buffers are kept cleared when DW\_apb\_i2c is not enabled.

Note

When a DW\_apb\_i2c Master is configured with IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 0, then for multiple I2C transfers, perform additional writes to the Tx FIFO such that the Tx FIFO does not become empty during the I2C transaction. If the Tx FIFO is completely emptied at any stage, then further writes to the Tx FIFO results in an independent I2C transaction.

#### 15.2.9.2.2 Dynamic IC\_TAR or IC\_10BITADDR\_MASTER Update

The DW\_apb\_i2c supports dynamic updating of the IC\_TAR (bits 9:0) and IC\_10BITADDR\_MASTER (bit 12) bit fields of the IC\_TAR register. In order to perform a dynamic update of the IC\_TAR register, the I2C\_DYNAMIC\_TAR\_UPDATE configuration parameter must be set to Yes (1). You can dynamically write to the IC\_TAR register provided the software ensures that there are no other commands in the Tx FIFO that use the existing TAR address. If the software does not ensure this, then IC\_TAR should be re-programmed only if the following conditions are met:

- DW\_apb\_i2c is not enabled (IC\_ENABLE[0]=0); OR DW\_apb\_i2c is enabled (IC\_ENABLE[0]=1); AND DW\_apb\_i2c is NOT engaged in any Master (tx, rx) operation (IC\_STATUS[5]=0); AND DW\_apb\_i2c is enabled to operate in Master mode (IC\_CON[0]=1); AND there are NO entries in the Tx FIFO (IC\_STATUS[2]=1); You can change the TAR address dynamically without losing the bus, only if the following conditions are met.
- DW\_apb\_i2c is enabled (IC\_ENABLE[0]=1); AND IC\_EMPTYFIFO\_HOLD\_MASTER\_EN configuration parameter is set to 1; AND DW\_apb\_i2c is enabled to operate in Master mode (IC\_CON[0]=1); AND there are NO entries in the Tx FIFO and the master is in HOLD state (IC\_INTR\_STAT[13]=1);1

**Note**

DW\_apb\_i2c uses the TAR address if either of the following conditions is true:

- The command has either RESTART or STOP bit set.
- The direction is changed in commands with a read command following a write command or vice versa

The updated TAR address comes into effect only when the next START or RESTART occurs on the bus.

#### 15.2.9.2.3 Master Transmit and Master Receive

The DW\_apb\_i2c supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (IC\_DATA\_CMD). The CMD bit [8] should be written to 0 for I2C write operations. Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the IC\_DATA\_CMD register, and a 1 should be written to the CMD bit. The DW\_apb\_i2c master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty-depending on the value of IC\_EMPTYFIFO\_HOLD\_MASTER\_EN, the master either inserts a STOP condition after completing the current transfers, or it checks to see if IC\_DATA\_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO. For more details, refer to "Tx FIFO Management and START, STOP and RESTART Generation".

**Note**

Master Receiver Mode is not supported in Ultra-Fast Mode (IC\_ULTRA\_FAST\_MODE=1) as Master Read transfers are not supported.

#### 15.2.9.3 Disabling DW\_apb\_i2c

The register IC\_ENABLE\_STATUS is added to allow software to unambiguously determine when the hardware has completely shutdown in response to bit 0 of the IC\_ENABLE register being set from 1 to 0. 1. If the software or application is aware the the DW\_apb\_i2c is not using the TAR address for the pending commands in the Tx FIFO, then it is possible to update the TAR address even while the Tx FIFO has entries (IC\_STATUS[2]= 0).

Only one register is required to be monitored, as opposed to monitoring two registers (IC\_STATUS and IC\_RAW\_INTR\_STAT) which is a requirement for DW\_apb\_i2c versions 1.05a or earlier.

**Note**

When IC\_EMPTYFIFO\_HOLD\_MASTER\_EN = 1, the DW\_apb\_i2c Master can be disabled only if the current command being processed-when the ic\_enable de-assertion occurs-has the STOP bit set to 1. When an attempt is

made to disable the DW\_apb\_i2c Master while processing a command without the STOP bit set, the DW\_apb\_i2c Master continues to remain active, holding the SCL line low until a new command is received in the Tx FIFO. When IC\_EMPTYFIFO\_HOLD\_MASTER\_EN =1 and the DW\_apb\_i2c Master is processing a command without the STOP bit set, you can issue the ABORT (IC\_ENABLE[1]) to relinquish the I2C bus and then disable DW\_apb\_i2c.

#### 15.2.9.3.1 Procedure

1. Define a timer interval (ti2c\_poll) equal to the 10 times the signaling period for the highest I2C transfer speed used in the system and supported by DW\_apb\_i2c. For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c\_poll is 25us.
2. Define a maximum time-out parameter, MAX\_T\_POLL\_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
3. Execute a blocking thread/process/function that prevents any further I2C master transactions to be started by software, but allows any pending transfers to be completed.

Note

This step can be ignored if DW\_apb\_i2c is programmed to operate as an I2C slave only.

4. The variable POLL\_COUNT is initialized to zero.
5. Set bit 0 of the IC\_ENABLE register to 0.
6. Read the IC\_ENABLE\_STATUS register and test the IC\_EN bit (bit 0). Increment POLL\_COUNT by one. If POLL\_COUNT >= MAX\_T\_POLL\_COUNT, exit with the relevant error code.
7. If IC\_ENABLE\_STATUS[0] is 1, then sleep for ti2c\_poll and proceed to the previous step. Otherwise, exit with a relevant success code.

#### 15.2.9.4 Aborting I2C Transfers

The ABORT control bit of the IC\_ENABLE register allows the software to relinquish the I2C bus before completing the issued transfer commands from the Tx FIFO. In response to an ABORT request, the controller issues the STOP condition over the I2C bus, followed by Tx FIFO flush. Aborting the transfer is allowed only in master mode of operation.

#### 15.2.9.4.1 Procedure

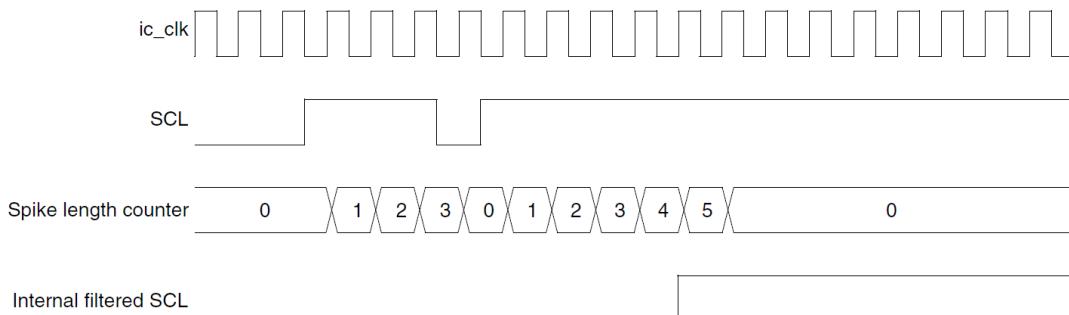
1. Stop filling the Tx FIFO (IC\_DATA\_CMD) with new commands.
2. When operating in DMA mode, disable the transmit DMA by setting TDMAE to 0.
3. Set bit 1 of the IC\_ENABLE register (ABORT) to 1.
4. Wait for the M\_TX\_ABRT interrupt.
5. Read the IC\_TX\_ABRT\_SOURCE register to identify the source as ABRT\_USER\_ABRT.

### 15.2.9.5 Spike Suppression

The DW\_apb\_i2c contains programmable spike suppression logic that match requirements imposed by the I2C Bus Specification for SS/FS (tSP, Table 9), HS (tSP, Table 11), and UFm (tSP, Table 13) modes. This logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of ic\_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic\_clk cycles can be programmed by the user and should be calculated taking into account the frequency of ic\_clk and the relevant spike length specification. Each counter is started whenever its input signal changes its value. Depending on the behavior of the input signal, one of the following scenarios occurs:

- The input signal remains unchanged until the counter reaches its count limit value. When this happens, the internal version of the signal is updated with the input value, and the counter is reset and stopped. The counter is not restarted until a new change on the input signal is detected.
- The input signal changes again before the counter reaches its count limit value. When this happens, the counter is reset and stopped, but the internal version of the signal is not updated. The counter remains stopped until a new change on the input signal is detected.

The timing diagram in Figure 15.26 illustrates the behavior described above.



**Figure 15.26:** Spike Suppression Example

The count limit value used in this example is 5 and was calculated for a 10 ns ic\_clk period and for SS/FS operation (50 ns spike suppression).

#### Note

There is a 2-stage synchronizer on the SCL input, but for the sake of simplicity this synchronization delay was not included in the timing diagram in Figure 15.26.

The I2C Bus Specification calls for different maximum spike lengths according to the operating mode-50 ns for SS and FS; 10 ns for HS, 10 ns for UFm, so three registers are required to store the values needed for each case:

- Register IC\_FS\_SPKLEN holds the maximum spike length for SS and FS modes
- Register IC\_HS\_SPKLEN holds the maximum spike value for HS mode.
- Register IC\_UFM\_SPKLEN holds the maximum spike value for UFm.

#### Note

- IC\_HS\_SPKLEN is implemented only if the component is configured for HS operation; that is, (IC\_MAX\_SPEED = High).
- IC\_UFM\_SPKLEN is implemented only if the component is configured for Ultra-Fast mode; that is, (IC\_ULTRA\_FAST\_MODE = High).
- IC\_FS\_SPKLEN and IC\_HS\_SPKLEN are not implemented when configured for Ultra-Fast mode; that is, (IC\_ULTRA\_FAST\_MODE=1).

These registers are 8 bits wide and accessible through the APB interface for read and write purposes; however, they can be written to only when the DW\_apb\_i2c is disabled. The minimum value that can be programmed into these registers is 1; attempting to program a value smaller than 1 results in the value 1 being written. The default value for these registers is automatically calculated in coreConsultant based on the value of ic\_clk period, but this value can be overridden by the user when configuring the component.

#### Note

- Because the minimum value that can be programmed into the IC\_FS\_SPKLEN, IC\_HS\_SPKLEN, and IC\_UFM\_SPKLEN registers is 1, the spike length specification can be exceeded for low frequencies of ic\_clk. Consider the simple example of a 10 MHz (100 ns period) ic\_clk; in this case, the minimum spike length that can be programmed is 100 ns, which means that spikes up to this length are suppressed.
- Standard synchronization logic (two flip-flops in series) is implemented upstream of the spike suppression logic and is not affected in any way by the contents of the spike length registers or the operation of the spike suppression logic; the two operations (synchronization and spike suppression) are completely independent. Because the SCL and SDA inputs are asynchronous to ic\_clk, there is one ic\_clk cycle uncertainty in the sampling of these signals; that is, depending on when they occur relative to the rising edge of ic\_clk, spikes of the same original length might show a difference of one ic\_clk cycle after being sampled.
- Spike suppression is symmetrical; that is, the behavior is exactly the same for transitions from 0 to 1 and from 1 to 0.

### 15.2.10 Fast Mode Plus Operation

In fast mode plus, the DW\_apb\_i2c allows the fast mode operation to be extended to support speeds up to 1000 Kb/s. To enable the DW\_apb\_i2c for fast mode plus operation, perform the following steps before initiating any data transfer:

1. Configure the Maximum Speed mode of DW\_apb\_i2c Master or Slave to Fast Mode or High Speed mode (IC\_MAX\_SPEED\_MODE> = 2).
2. Set ic\_clk frequency greater than or equal to 32 MHz (refer to "Standard Mode (SM), Fast Mode (FM), and Fast Mode Plus (FM+) with IC\_CLK\_FREQ\_OPTIMIZATION = 0").
3. Program the IC\_CON register [2:1] = 2'b10 for fast mode or fast mode plus.
4. Program IC\_FS\_SCL\_LCNT and IC\_FS\_SCL\_HCNT registers to meet the fast mode plus SCL (refer to "IC\_CLK Frequency Configuration").
5. Program the IC\_FS\_SPKLEN register to suppress the maximum spike of 50ns.
6. Program the IC\_SDA\_SETUP register to meet the minimum data setup time (tSU; DAT).

### 15.2.11 Bus Clear Feature

DWC\_apb\_i2c supports the bus clear feature that provides graceful recovery of data (SDA) and clock (SCL) lines during unlikely events in which either the clock or data line is stuck at LOW. The following sections describes the SDA and SCL lines stuck at LOW recovery mechanisms:

- "SDA Line Stuck at LOW Recovery"
- "SCL Line is Stuck at LOW"

#### 15.2.11.1 SDA Line Stuck at LOW Recovery

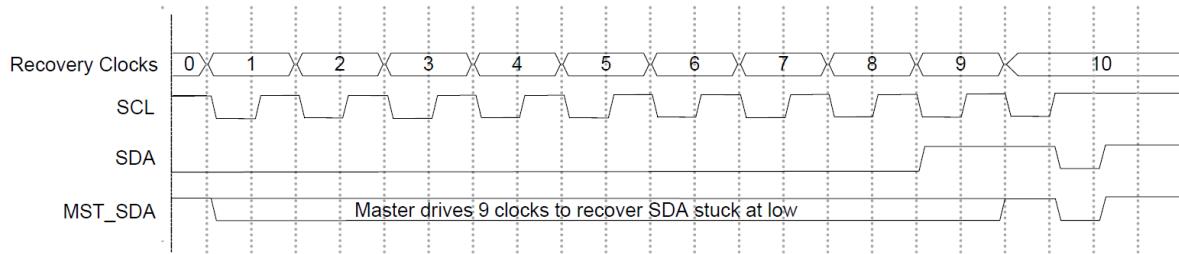
In case of SDA line stuck at LOW, the master performs the following actions to recover as shown in Figure 15.27 and Figure 15.28:

1. Master sends a maximum of 9 clock pulses to recover the bus LOW within those 9 clocks.
  - The number of clock pulses will vary with the number of bits that remain to be sent by the slave.

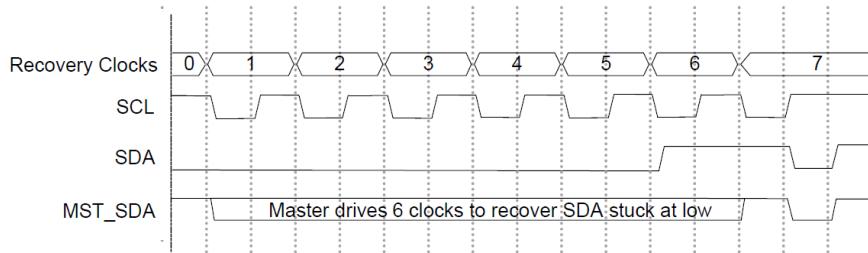
As the maximum number of bits is 9, master sends up to 9 clock pluses and allows the slave to recover it.

  - The master attempts to assert a Logic 1 on the SDA line and check whether SDA is recovered. If the SDA is not recovered, it will continue to send a maximum of 9 SCL clocks.
2. If SDA line is recovered within 9 clock pulses then the master will send the STOP to release the bus.
3. If SDA line is not recovered even after the 9th clock pulse then system needs a hardware reset.

The detailed flow to recover the SDA stuck at LOW is explained in the section "Programming Flow for SCL and SDA Bus Recovery" .



**Figure 15.27: SDA Recovery with 9 SCL Clocks**



**Figure 15.28:** SDA Recovery with 6 SCL Clocks

### 15.2.11.2 SCL Line is Stuck at LOW

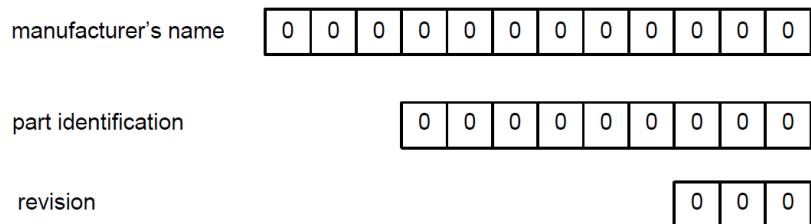
In the unlikely event (due to an electric failure of a circuit) where the clock (SCL) is stuck to LOW, there is no effective method to overcome this problem but to reset the bus using the hardware reset signal. The detailed flow to recover the SCL stuck at LOW is explained in "Programming Flow for SCL and SDA Bus Recovery".

### 15.2.12 Device ID

A Device ID field is an optional 3-byte read-only (24 bits) word, which provides the following information:

- Twelve bits with the manufacturer's name, which is unique for every manufacturer.
- Nine bits with the part identification, which is assigned by the manufacturer.
- Three bits with the die revision, which is assigned by the manufacturer.

Figure 15.29 shows the Device ID field structure.



**Figure 15.29:** Device ID Field Structure

For reading the Device ID of a particular slave, the master can follow the procedure in "Programming Flow for Reading the Device ID". The Device ID that is read will be available in RX FIFO, which can be read using IC\_DATA\_CMD

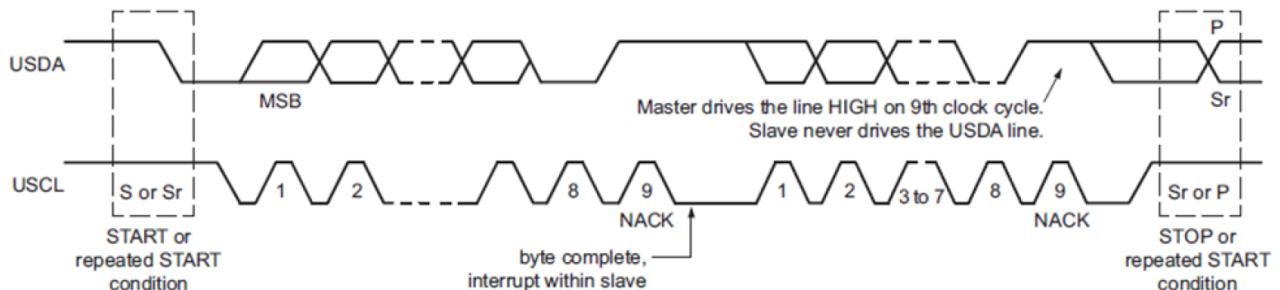
register. In case of a slave, the user has to configure the Device ID using the IC\_DEVICE\_ID\_VALUE coreConsultant parameter and user can read the Device ID of the slave using IC\_DEVICE\_ID register.

#### Note

Device ID is not supported for 10-bit addressing and High Speed transfers (HS mode).

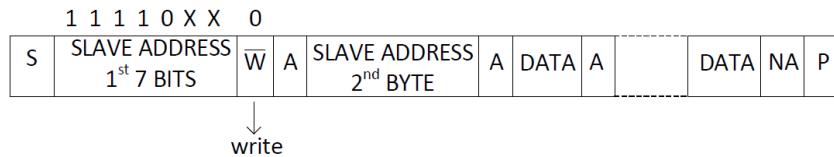
### 15.2.13 Ultra-Fast Speed Mode

The Ultra-Fast Speed mode is a variant of I2C Bus Speed mode that operates from DC (0) to 5 MHz transmitting data in one direction. It is useful for speeds greater than 1 MHz to drive LED controllers and other gaming systems that do not need feedback. Ultra-Fast speed mode is based on the standard I2C Protocol, which consists of START, slave address, command bit, ninth clock (ACK cycle) and a STOP bit. The command bit should be always 'write' (0) only since it is a unidirectional bus (except for the START byte). The data bit on the ninth (ACK) cycle is driven high by the master, ignoring the ACK cycle due to unidirectional nature of bus. The driver used for Ultra-Fast Mode is push-pull driver. The Master consists of serial clock (ic\_clk\_oe, USCL) and a serial data (ic\_data\_oe, USDA) output signals. The Output signals are Active-Low in nature. The Slave consists of serial clock (ic\_clk\_in\_a, USCL) and serial data (ic\_data\_in\_a, USDA) input signals. The input signals are Active-High in nature. The UFm I2C-bus does not have the multi-master capability and hence, it does not consist of wired-AND open-drain driver. In the UFm I2C bus, the master is the only device that initiates a data transfer (write transfer) on the bus and provides the clock signals to support that transfer. All other devices are considered as slaves. Because of single master support, the arbitration, synchronization, clock stretching mechanisms are not applicable. The Byte format, START and STOP generation are same as in other modes of the I2C Protocol except for the ignorance of ACK cycle. The Slave never drives anything on the bus hence, the master always drives NACK during the ninth cycle of the transfer as shown in Figure 15.30.



**Figure 15.30:** UFm-I2C Byte Transfer

In UFm-I2C mode, the slave is not allowed to hold the clock LOW if it cannot receive another complete byte of data or while it is performing some other function, for example, servicing an internal interrupt. The ninth clock cycle that represents ACK/NACK of the byte is not applicable as slave will not respond and it is preserved in UFm to be compatible with the I2C Protocol. The 8th bit of the address that represents Read or write transfer should be always set to write (0), since Read is not supported in UFm (except for the START Byte). The Combined format of I2C Protocol is not supported in UFm-I2C mode. The 10-bit addressing that expands the number of possible devices is supported in UFm-I2C mode and it behaves similar to other modes as shown in Figure 15.31 (Only write transfer is supported).



**Figure 15.31:** 10-bit addressing write transfer

The UFm-I2C mode supports START byte and general call features similar to other I2C modes. If the slave is not responsive (determined through external feedback and not through UFm I2C-bus), then the slave can reset through software reset or external hardware reset.

### 15.2.14 SMBus/PMBus

The SMBus is designed to provide a predictable communication line between a system and its devices. It describes the Device timeout definitions and their conditions.

#### 15.2.14.1 tTimeout,MIN Parameter

This Parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. It is highly recommended that a slave device release the bus (stop driving the bus and let SMBCLK and SMBDAT float high) when it detects any single clock held low longer than tTIMEOUT,MIN. Devices that have detected this condition must reset their communication interface and be able to receive a new START condition in no later than tTIMEOUT,MAX. The DW\_apb\_i2c enables the Bus clear feature in SMBus mode and the user can use the IC\_SCL\_STUCK\_TIMEOUT Register to program the tTIMEOUT,MIN Value to detect the SMBCLK low timeout. The DW\_apb\_i2c slave device will reset its communication interface and release both SCL and SDA lines after detecting the SCL\_STUCK\_TIMEOUT interrupt. The DW\_apb\_i2c master has a provision to generate the Abort which completes the current transfer and generate STOP condition on the bus through programming the IC\_ENABLE[1] register bit.

#### 15.2.14.2 Master Device Clock Extension

The interval tLOW: MEXT is defined as the cumulative time a master device is allowed to extend its clock cycles within one byte in a message as measured from:

- START to ACK
- ACK to ACK
- ACK to STOP.

The DW\_apb\_i2c Master uses the IC\_SMBUS\_CLOCK\_LOW\_MEXT register to detect the Master device clock extension timeout and generates SMBUS\_CLK\_LOW\_MEXT interrupt.

#### 15.2.14.3 Slave Device Clock Extension

The interval tLOW:SEXT is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. The DW\_apb\_i2c Master uses the IC\_SMBUS\_CLOCK\_LOW\_SEXT register to detect the Slave device clock extension timeout and generates SMBUS\_CLK\_LOW\_SEXT interrupt. A Master is allowed to abort the transaction in progress to any slave that violates the tLOW:SEXT or tTIMEOUT,MIN specifications through the enabling the user abort (IC\_ENABLE[1]).

#### 15.2.14.4 SMBDAT Low Timeout

A malfunctioning device holds the SMBDAT line low indefinitely. This would prevent the master from issuing a STOP condition and ending a transaction. If SMBDAT is still low tTIMEOUT,MAX after SMBCLK has gone high at the end of a transaction the master should hold SMBCLK low for at least tTIMEOUT,MAX in an attempt to reset the SMBus interface of all of the devices on the bus. The DW\_apb\_i2c enables the Bus clear feature in SMBus mode and the user can use the IC\_SDA\_STUCK\_TIMEOUT Register to program the SMBDAT timeout value to detect the SMBDAT low timeout. If SMBDAT line is stuck at low, the SDA\_STUCK\_TIMEOUT abort is generated and software can enable the SMBUS\_CLK\_RESET register bit of IC\_ENABLE register to hold the SCL low for IC\_SCL\_STUCK\_TIMEOUT which in turn resets the SMBus interface of all devices on the bus.

#### 15.2.14.5 Bus Protocols

A typical SMBus device will have a set of commands by which data can be read and written. All commands are one byte long while their arguments and return values can vary in length. In accordance with the SMBus specification, the most significant bit (MSB) is transferred first. There are eleven possible command protocols for any given device. These commands are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write, and Block Write-Block Read Process Call. SMBus protocols for message transactions are generally different from I2C data transfer commands. It is still possible to program an SMBus master to deliver I2C data transfer commands. The following table describes the derivation of SMBus Bus Protocols through Tx-FIFO commands in DW\_apb\_i2c. In the SMBus Master mode, all the receive data bytes will be available in Rx-FIFO. In the SMBus Slave mode, all the bus protocol command codes and data bytes will received in the Rx-FIFO and read request data bytes must be sent using Tx-FIFO, similar to the I2C mode.

Protocol	Required TxFIFO Commands	Command/Data (IC_DATA_CMD[7:0])	CMD bit (IC_DATA_CMD[8])	STOP bit (IC_DATA_CM D[9])	Remarks
Quick Command	1	Not Applicable	Set the command [R/W]	Set to 1	Set IC_TAR[11] and IC_TAR[16] to 1
Send Byte	1	Data Byte	Set to 0	Set to 1	
Receive Byte	1	Not Applicable	Set to 1	Set to 1	
Write Byte	2	Command Code	Set to 0	Set to 0	
Write Byte	2	Data Byte	Set to 0	Set to 1	
Write Word	3	Command Code	Set to 0	Set to 0	

Table 15.2: SMBus Bus Protocols Usage in DW\_apb\_i2c (Continued on next page)

continued from previous page					
Write Word	3	Data Byte Low	Set to 0	Set to 0	
Write Word	3	Data Byte High	Set to 0	Set to 1	
Read Byte	2	Command Code	Set to 0	Set to 0	
Read Byte	2	Not Applicable	Set to 1	Set to 1	
Read Word	3	Command Code	Set to 0	Set to 0	
Read Word	3	Not Applicable	Set to 1	Set to 0	
Read Word	3	Not Applicable	Set to 1	Set to 1	
Process Call	5	Command Code	Set to 0	Set to 0	
Process Call	5	Data Byte Low	Set to 0	Set to 0	
Process Call	5	Data Byte High	Set to 0	Set to 0	
Process Call	5	Not Applicable	Set to 1	Set to 0	
Process Call	5	Not Applicable	Set to 1	Set to 1	
Block Write	N+2	Command Code	Set to 0	Set to 0	
Block Write	N+2	Data Byte	Set to 0	Set to 0	
Block Write	N+2	N+1) Data Byte N	Set to 0	Set to 1	
Block Read	N+2	Command Code	Set to 0	Set to 0	
Block Read	N+2	Not Applicable	Set to 0	Set to 0	
Block Read	N+2	N+1) Not Applicable	Set to 0	Set to 1	
Block Write-Block Read Process Call	M+N+2	Command Code	Set to 0	Set to 0	
Block Write-Block Read Process Call	M+N+2	Data Byte 1	Set to 0	Set to 0	
Block Write-Block Read Process Call	M+N+2	M+1) Data Byte M	Set to 0	Set to 0	
Block Write-Block Read Process Call	M+N+2	M+2) Not Applicable	Set to 1	Set to 0	

Table 15.2: SMBus Bus Protocols Usage in DW\_apb\_i2c (Continued on next page)

continued from previous page					
Block Write-Block Read Process Call	M+N+2	M+3) Not Applicable	Set to 1	Set to 0	
Block Write-Block Read Process Call	M+N+2	M+N+1) Not Applicable	Set to 1	Set to 1	
SMBUS Host Notify Protocol	3	Device-Address	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Host Address (0001 000)
SMBUS Host Notify Protocol	3	Data Byte Low	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Host Address (0001 000)
SMBUS Host Notify Protocol	3	Data Byte High	Set to 0	Set to 1	Set IC_TAR[6:0] to SMB Host Address (0001 000)

**Table 15.2:** SMBus Bus Protocols Usage in DW\_apb\_i2c

DW\_apb\_i2c Slave can be enabled to receive only Quick command through enabling the SLAVE\_QUICK\_CMD\_EN bit in the IC\_CON Register. Whenever this bit is selected the slave only receives quick commands and will not accept other Bus Protocols. The DW\_apb\_i2c slave issues the SMBUS\_QUICK\_DET interrupt upon receiving the QUICK command. SMBus introduces a Packet Error checking Mechanism through appending PEC Byte at the end of the Bus Protocol. This can be achieved through adding an extra command (PEC byte) while transferring and decoding it while receiving by the software.

#### 15.2.14.6 SMBUS Address Resolution Protocol

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device by the Host. This feature allows the devices to be 'hot-plugged' in to the system. SMBus introduces a 128-bit Unique Device ID (UDID) for each device in the system to isolate each device for the purpose of address assignment. DW\_apb\_i2c uses the IC\_SMBUS\_UDID\_MSB parameter for upper constant 96 bits and 'IC\_SMBUS\_ARP\_UDID\_LSB' register for lower variable 32 bits of the UDID. DW\_apb\_i2c uses the PERSISTANT\_SLV\_ADDR\_EN register bit in IC\_CON register to indicate whether the DW\_apb\_i2c supports persistent slave address. DW\_apb\_i2c master can issue general and directed Address Resolution Protocol (ARP) commands to assign the dynamic address for the slaves in the SMBus system. Table 15.3 describes the derivation of SMBus ARP commands through Tx-FIFO commands in DW\_apb\_i2c.

ARP Command	Required Tx_FIFO Commands	Command/Data (IC_DATA_CMD[7:0])	CMD Bit (IC_DATA_CMD[8])	STOP bit (IC_DATA_CMD[9])	Remarks
Prepare for ARP	2	Command = '0000 0001'	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Default Address (1100 001)
Prepare for ARP	2	PEC Byte	Set to 0	Set to 1	Set IC_TAR[6:0] to SMB Default Address (1100 001)
Reset Device (General)	2	Command = '0000 0010'	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Default Address (1100 001)
Reset Device (General)	2	PEC Byte	Set to 0	Set to 1	Set IC_TAR[6:0] to SMB Default Address (1100 001)
Get UDID (General)	20	Command = '0000 0011'	Set to 0	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID bytes. 3. Last read command for the slave address.
Get UDID (General)		Not Applicable	Set to 1	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID bytes. 3. Last read command for the slave address.
Get UDID (General)		Not Applicable	Set to 1	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID bytes. 3. Last read command for the slave address.
Get UDID (General)		Not Applicable	Set to 1	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID bytes. 3. Last read command for the slave address.

Table 15.3: Derivation of SMBus ARP Command Through TxFIFO Commands (Continued on next page)

--	--	--	--	--	--

continued from previous page

Get UDID (General)		PEC Byte	Set to 1	Set to 1	<ol style="list-style-type: none"> <li>1. Set IC_TAR[6:0] to SMB Default Address (1100 001).</li> <li>2. 16 Reads to be performed for the 128 UDID bytes.</li> <li>3. Last read command for the slave address.</li> </ol>
Assign Address	20	Command = '0000 0011'	Set to 0	Set to 0	<ol style="list-style-type: none"> <li>1. Set IC_TAR[6:0] to SMB Default Address (1100 001).</li> <li>2. 16 Writes to be performed for the 128 UDID byte.</li> <li>3. Last Write command for the Assigned slave address.</li> </ol>
Assign Address	20	Byte Count = 17	Set to 0	Set to 0	<ol style="list-style-type: none"> <li>1. Set IC_TAR[6:0] to SMB Default Address (1100 001).</li> <li>2. 16 Writes to be performed for the 128 UDID byte.</li> <li>3. Last Write command for the Assigned slave address.</li> </ol>
Assign Address	20	UDID Byte 15	Set to 0	Set to 0	<ol style="list-style-type: none"> <li>1. Set IC_TAR[6:0] to SMB Default Address (1100 001).</li> <li>2. 16 Writes to be performed for the 128 UDID byte.</li> <li>3. Last Write command for the Assigned slave address.</li> </ol>
Assign Address	20	UDID Byte 14	Set to 0	Set to 0	<ol style="list-style-type: none"> <li>1. Set IC_TAR[6:0] to SMB Default Address (1100 001).</li> <li>2. 16 Writes to be performed for the 128 UDID byte.</li> <li>3. Last Write command for the Assigned slave address.</li> </ol>

**Table 15.3:** Derivation of SMBus ARP Command Through TxFIFO Commands (Continued on next page)

continued from previous page					
------------------------------	--	--	--	--	--

Assign Address	20	Assigned Address	Set to 0	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Writes to be performed for the 128 UDID byte. 3. Last Write command for the Assigned slave address.
Assign Address	20	PEC Byte	Set to 01	Set to 1	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Writes to be performed for the 128 UDID byte. 3. Last Write command for the Assigned slave address.
Get UDID (Directed)	19	Command = '0000 0011'	Set to 0	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID byte. 3. Last Read command for the slave address.
Get UDID (Directed)	19	Slave address[6:0],1}	Set to 1	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID byte. 3. Last Read command for the slave address.
Get UDID (Directed)	19	Not Applicable	Set to 1	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID byte. 3. Last Read command for the slave address.

**Table 15.3:** Derivation of SMBus ARP Command Through TxFIFO Commands (Continued on next page)

continued from previous page					
------------------------------	--	--	--	--	--

Get UDID (Directed)	19	Not Applicable	Set to 1	Set to 0	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID byte. 3. Last Read command for the slave address.
Get UDID (Directed)	19	PEC Byte	Set to 1	Set to 1	1. Set IC_TAR[6:0] to SMB Default Address (1100 001). 2. 16 Reads to be performed for the 128 UDID byte. 3. Last Read command for the slave address.
Reset Device (Directed)	2	command = slave address[6: 0],0	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Default Address (1100 001)
Reset Device (Directed)	2	PEC byte	Set to 0	Set to 1	Set IC_TAR[6:0] to SMB Default Address (1100 001)
Notify ARP Master	3	Device Address = '1100 0010'	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Host Address (0001 000)
Notify ARP Master	3	Data Byte Low = '0000 0000'	Set to 0	Set to 0	Set IC_TAR[6:0] to SMB Host Address (0001 000)
Notify ARP Master	3	Data Byte High = '0000 0000'	Set to 0	Set to 1	Set IC_TAR[6:0] to SMB Host Address (0001 000)

**Table 15.3:** Derivation of SMBus ARP Command Through TxFIFO Commands

#### Note

- DW\_apb\_i2c slave hardware: - Handles the generation, detection, and NACKing of the wrong PEC ( $C(X)=X8+X2+X1+1$ ) for the ARP Commands. - Does not handle the PEC for Non-ARP commands.
- DW\_apb\_i2c master hardware does not handle PEC for both APR and non- ARP commands.

#### 15.2.14.6.1 Procedure to Perform ARP in Master Mode

To use the DW\_apb\_i2c as a SMBus Master/Host for assigning the unique address to each slave device to resolve the slave address conflicts, perform the following steps:

1. After a reset or a cold power up, the SMBus host or master issues a "Prepare to ARP" command to indicate that the master is carrying an ARP to assign dynamic addresses to all devices. Slave must flush any pending host notify commands.

2. An acknowledgement received for the "Prepare to ARP" command indicates that ARP-capable devices exist in the system and the "Get UDID" command must be issued. A NACK indicates that ARP-capable devices do not exist or currently all slaves have their addresses resolved. In this case, the master must complete steps outlined from Step 8 onwards. The DW\_apb\_i2c master indicates NACK reception through 'ABRT\_7B\_ADDR\_NOACK' and 'ABRT\_TXDATA\_NOACK' bits of IC\_TX\_ABRT\_SOURCE register.
3. DW\_apb\_i2c Master issues 'Get UDID' to receive the UDID information of the slave for assigning the dynamic address.
4. If the first three bytes of the "Get UDID" command are ACK'ed and the receive byte count is 0x11, then the master issues the "Assign Address" command. Else, the master must complete steps outlined in step 8 onwards to indicate that the ARP is complete. DW\_apb\_i2c Master indicates NACK reception through ABRT\_7B\_ADDR\_NOACK and ABRT\_TXDATA\_NOACK bits of the IC\_TX\_ABRT\_SOURCE register.
5. The Master issues the "Assign Address" command to assign the Dynamic address to the slave whose UDID is received through "Get UDID command".
6. If the assigned address packet is ACK'ed, then Master removes the assigned address from the address pool and moves to Step 3 to get UDID of another slave. If the packet is not ACK'ed, then master will not remove the address from the address pool and moves to Step 3 to get UDID of same slave or another slave.
7. If the Assign Address is ACK'ed, then Master stores the assigned address in the used address pool with the UDID characteristics of the device.
8. The Master moves to Step 3 to issue a 'Get UDID' command again to receive the UDID of another slave. If it receives NACK for 'Get UDID', the Master moves to Step 9.
9. The DW\_apb\_i2c can be switched to Slave mode to detect device requests for Host Notify Protocol.
10. If the DW\_apb\_i2c switched to slave mode and DW\_apb\_i2c detects the Host Notify Protocol, then this indicates that a slave is requesting for the dynamic address and the Master has to undergo the ARP as outlined in Step 11.
11. If the DW\_apb\_i2c is in Master mode, then move to Step 3 for performing ARP procedure, otherwise move to Step 12.
12. The DW\_apb\_i2c is switched to Master Mode and moves to Step 3 to perform ARP procedure.

The detailed flow diagram is explained in Figure 7-10 .

#### 15.2.14.6.2 Procedure to Perform ARP in Slave Mode

The DW\_apb\_i2c as a SMBus Slave performs the following tasks:

- Decodes the ARP commands and responds based on internal state flags SMBUS\_SLAVE\_ADDR\_VALID and 'SMBUS\_SLAVE\_ADDR\_RESOLVED' of the IC\_STATUS register.
- Generates and Validates the PEC byte of ARP commands
- Generates ACK for the PEC byte only if it matches the CRC value calculated on data it received. If not, NACK the PEC byte.

When another SMBus Master/Host device on the bus generates the ARP commands and requests to participate in the ARP, the DW\_apb\_i2c acts as a SMBus slave and performs the following steps:

1. After a reset or a cold power up, the DW\_apb\_i2c slave device checks whether it supports a persistent slave address.
2. If DW\_apb\_i2c has a persistent slave address (PSA), which is indicated by the Address Valid flag being set, then PSA is set in the Slave Address Register (IC\_SAR) register. If the flag is not set, then proceed to Step 4.

3. DW\_apb\_i2c persistent slave stores the persistent address in IC\_SAR and sets Address Valid flag to 1 and Address Resolved Flag to 0.
4. DW\_apb\_i2c Non Persistent slave (non-PSA) clears both Address Valid and Address Resolved Flags.
5. DW\_apb\_i2c Checks whether any Packet received has ARP Default address in the slave address field of the packet to decide on ARP command or normal command. If there is a match then DW\_apb\_i2c slave proceeds to Step 6, otherwise to Step 25.
6. If DW\_apb\_i2c detects a packet addressed to the SMBus Device Default Address, it checks the command field to determine if this is the "Prepare to ARP" command. If so, then it proceeds to Step 7, otherwise it proceeds to Step 8.
7. Upon receipt of the "Prepare to ARP" command, the DW\_apb\_i2c acknowledges the packet and clears the Address Resolved flag in order to participate in the ARP Process. DW\_apb\_i2c proceeds to Step 5 and waits for another SMBus Packet.
8. The DW\_apb\_i2c checks the command field to verify if the "Reset Device" command was issued. If yes, the DW\_apb\_i2c proceeds to Step 9, otherwise it proceeds to Step 10.
9. Upon receipt of the "Reset Device" command, the DW\_apb\_i2c acknowledges the packet and clears the Address Resolved and Address Valid (If non-PSA and ic\_con[19]=0) flags. DW\_apb\_i2c proceeds to Step 5 and waits for another SMBus Packet.
10. The device checks the command to verify if the "Assign Address" command was issued. If yes, then it proceeds to Step 11, otherwise proceeds to Step 13.
11. Upon receipt of the "Assign Address" command, the DW\_apb\_i2c compares its UDID with one its received bytes. If any byte does not match, then DW\_apb\_i2c will not acknowledge that byte and subsequent bytes also. If all bytes in the UDID matches, then the DEVICE proceeds to Step 12, otherwise it proceeds to Step 5 and waits for another SMBus packet.
12. After the UDID is matched in Step 11, the DW\_apb\_i2c will receive the slave address and sets the IC\_SAR register with this slave address. The DW\_apb\_i2c sets its Address Valid and Address Resolved flags, which means it has received the dynamic address and will no longer respond to the "Get UDID" command unless it receives the "Prepare to ARP" or "Reset Device" commands. DW\_apb\_i2c now proceeds to Step 5 and waits for another SMBus packet.
13. The DW\_apb\_i2c checks the command field to verify if the "Get UDID" command was issued. If yes, then it proceeds to Step 14, otherwise to Step 19.
14. Upon receipt of the "Get UDID" command, the DW\_apb\_i2c checks its Address Resolved flag to determine whether it must participate in an ARP process. If set, then its address has already been resolved by the ARP Master, so the device proceeds to Step 5 and waits for another SMBus packet. If the ARP Flag is cleared, then it proceeds to Step 15.
15. The DW\_apb\_i2c returns its UDID and monitors the SMBus data line for collisions. If a collision is detected at any time, DW\_apb\_i2c generates the SLV\_ARB\_LOST bit and stops transmitting. Further, it proceeds to Step 5 and waits for another SMBus packet. If collisions are not detected, then DW\_apb\_i2c proceeds to Step 16.
16. The DW\_apb\_i2c check its Address Valid (AV) flag to determine the value to return for the Device Slave Address field. If the AV flag is set, then it proceeds to Step 17, otherwise it proceeds to Step 18.
17. When the AV flag is set, the current IC\_SAR is valid, therefore the device returns this for the Device Slave Address field (with bit 0 set) and monitors the SMBus data line for collisions. DW\_apb\_i2c proceeds to Step 5 and waits for another SMBus Packet.
18. When the AV flag is not set, the current slave address (IC\_SAR) is invalid. Therefore, the DW\_apb\_i2c returns a value of FFH and monitors the SMBus data line for collisions. The device requires an address assignment if the ARP master receives the FFH value. DW\_apb\_i2c proceeds to Step 5 and waits for another SMBus packet.
19. The DW\_apb\_i2c may be receiving a directed command. If the Address Valid flag is set and address is the same as in IC\_SAR, then proceed to Step 20 otherwise, proceed to Step 5 to wait for another SMBus packet.

20. If the Address Valid flag is set, check if the command is a directed "Reset Device" command. If yes, then proceed to Step 21, otherwise proceed to Step 22.
21. Upon receipt of the "Reset Device" command, the DW\_apb\_i2c acknowledges the packet and clears the Address Resolved and Address Valid (If non-PSA and ic\_con[19]=0) flags. DW\_apb\_i2c proceeds to Step 5 and waits for another SMBus Packet.
22. DW\_apb\_i2c checks whether the received command is a "Directed Get UDID" command. If yes, then proceed to Step 23 and return the UDID information. If not, then proceed to Step 24.
23. If the received command is a "Directed Get UDID" command, then return the UDID information and current slave address, proceed to Step 5 and wait for another SMBus Packet.
24. If the received command is a "Directed Get UDID" command, the DW\_apb\_i2c has not received a valid ARP command and hence DW\_apb\_i2c NACKs the command and proceeds to Step 5 and wait for another SMBus Packet.
25. If the Address Valid bit is set then it proceeds to Step 26, otherwise it proceeds to Step 5 and waits for another SMBus Packet. The received address is not the SMBus Device Default Address and the packet may be addressed to the DW\_apb\_i2c's core function. The device checks its Address Valid bit to determine whether to respond.
26. When the address valid bit is set, DW\_apb\_i2c has a valid slave address. It compares the received slave address to its slave address, and if there is a match, DW\_apb\_i2c proceeds to Step 27, otherwise it proceeds to Step 5 and waits for another SMBus Packet.

The detailed flow diagram is explained in Figure 7-11 .

#### 15.2.14.7 SMBUS Additional Slave Address

DW\_apb\_i2c supports second optional slave address decode capability. It can be configured to contain an extra slave address register IC\_OPTIONAL\_SAR. If configured with this additional register, user can write any valid slave address to this register which will be matched against an incoming slave address on SMBus. A match of incoming address with either IC\_SAR register or IC\_OPTIONAL\_SAR register will cause DW\_apb\_i2c to acknowledge the transaction and respond to it accordingly. Use of this additional slave address register is controlled by OPTIONAL\_SAR\_CTRL (IC\_CON[17]) bit. If OPTIONAL\_SAR\_CTRL bit is programmed to be 1, then IC\_OPTIONAL\_SAR register will be used to match the incoming address. All restrictions of IC\_SAR register applies to IC\_OPTIONAL\_SAR register as well. The default value that IC\_OPTIONAL\_SAR register obtains after reset can be configured by the IC\_OPTIONAL\_SAR\_DEFAULT parameter.

#### 15.2.14.8 SMBUS Optional Signals

The SMBus standard supports these optional signals:

- SMBus Suspend Signal
- SMBus Alert Signal

As these signals are optional, DW\_apb\_i2c can be configured to include these signals through IC\_SMBUS\_SUSPEND\_ALERT parameter.

#### 15.2.14.8.1 SMBus Suspend Signal

The SMBus suspend signal (SMBSUS#) is an optional signal which is asserted by the system controller (mostly the Host) to indicate that the system should enter in low power suspend mode. It is output from the system controller and input to all other devices. This signal is an active low signal. DW\_apb\_i2c implements this functionality using following signals:

- ic\_smbsus\_in\_n
- ic\_smbsus\_out\_n

Output signal ic\_smbsus\_out\_n is controlled directly by the SMBUS\_SUS\_CTRL bit (IC\_ENABLE[17]). If this bit is programmed to 1, ic\_smbsus\_out\_n signal goes to 0 as soon as master finishes any ongoing transfer. For coming out of the suspend mode, user needs to clear this bit, which deasserts the ic\_smbsus\_out\_n signal. Input signal ic\_smbsus\_in\_n generates interrupt ic\_smbsus\_det\_intr (or ic\_smbsus\_det\_intr\_n) on the falling edge. This interrupt can be used by the software to enter the Low Power Mode. Current status of this ic\_smbsus\_in\_n can be read from SMBUS\_SUSPEND\_STATUS bit of IC\_STATUS (19) register.

#### 15.2.14.8.2 SMBus Alert Signal

The SMBus alert signal (SMBALERT#) is other optional signal specified by the SMBus standard. It can be used by simple devices to request the attention of the host. Devices can use the SMBALERT# signal to request the attention of the host with master functionality. This signal is input to host device and output from all other devices. Since multiple devices may implement SMBALERT#, it is required to be a wired-AND signal. Upon detecting a SMBALERT# signal, a host must send an alert response address which is acknowledge by alerting the device and it sends the address to the host and deasserts the alert signal. If host still detects an asserted alert signal, it repeats sending alert response address. DW\_apb\_i2c implements this functionality using following signals:

- ic\_smbalert\_in\_n
- ic\_smbalert\_oe

Output signal ic\_smbalert\_oe is open drain/open collector pull down driver and should be used similar to ic\_clk\_oe and ic\_data\_oe on a system implementation. Assertion of ic\_smbalert\_oe is controlled by SMBUS\_ALERT\_CTRL bit (IC\_ENABLE[18]). Once asserted by user, DW\_apb\_i2c waits for alert response address to be sent by master. Upon receiving it, contents of IC\_SAR[7:0] register are sent to the master. When successful, DW\_apb\_i2c clears the SMBUS\_ALERT\_CTRL bit and deasserts the ic\_smbalert\_oe signal. Input signal ic\_smbalert\_in\_n generates interrupt ic\_smbalert\_det\_intr (or ic\_smbalert\_det\_intr\_n) on falling edge. If working as host, user needs to service this interrupt by sending read byte command with Alert Response Address. Current status of ic\_smbalert\_in\_n can be read from SMBUS\_ALERT\_STATUS bit (IC\_STATUS[20])

### 15.2.15 IC\_CLK Frequency Configuration

When the DW\_apb\_i2c is configured as a Standard (SS), Fast (FS)/Fast-Mode Plus (FM+), or High Speed (HS) master, the \*CNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing. The \*CNT registers are:

- IC\_SS\_SCL\_HCNT
- IC\_SS\_SCL\_LCNT
- IC\_FS\_SCL\_HCNT
- IC\_FS\_SCL\_LCNT

- IC\_HS\_SCL\_HCNT
- IC\_HS\_SCL\_LCNT

When the DW\_apb\_i2c is configured as a Ultra-Fast Mode master, the \*CNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing. The \*CNT registers for this mode are:

- IC\_UFM\_SCL\_HCNT
- IC\_UFM\_SCL\_LCNT

#### Note

The tBUF timing and setup/hold time of START, STOP and RESTART registers uses \*HCNT/ \*LCNT register settings for the corresponding speed mode.

#### Note

It is not necessary to program any of the \*CNT registers if the DW\_apb\_i2c is enabled to operate only as an I2C slave, since these registers are used only to determine the SCL timing requirements for operation as an I2C master. Table 15.4 lists the derivation of I2C timing parameters from the \*CNT programming registers.

Timing Parameter	Symbol	Standard Speed	Fast Speed / Fast Speed Plus	High Speed (100 pf)	High Speed (400 pf)
LOW period of the SCL clock	tLOW	IC_SS_SCL_LCNT	IC_FS_SCL_LCNT	IC_HS_SCL_LCNT	IC_HS_SCL_LCNT
HIGH period of the SCL clock	tHIGH	IC_SS_SCL_HCNT	IC_FS_SCL_HCNT	IC_HS_SCL_HCNT	IC_HS_SCL_HCNT
Setup time for a repeated START condition	tSU; STA	IC_SS_SCL_LCNT	IC_FS_SCL_HCNT	IC_HS_SCL_LCNT / 2	(IC_HS_SCL_LCNT) / 2
Hold time (repeated) START condition*	tHD; STA	IC_SS_SCL_HCNT	IC_FS_SCL_HCNT	IC_HS_SCL_LCNT	(IC_HS_SCL_LCNT) / 2
Setup time for STOP condition	tSU; STO	IC_SS_SCL_HCNT	IC_FS_SCL_HCNT	IC_HS_SCL_LCNT	(IC_HS_SCL_LCNT) / 2
Bus free time between a STOP and a START condition	tBUF	IC_SS_SCL_LCNT	IC_FS_SCL_LCNT	NA	NA
Spike length	tSP	IC_FS_SP-KLEN	IC_FS_SPKLEN	IC_HS_SPKLEN	IC_HS_SPKLEN
Data hold time	tHD; DAT	IC_SDA_HOLD	IC_SDA_HOLD	IC_SDA_HOLD	IC_SDA_HOLD
Data setup time	tSU; DAT	IC_SDA_SETUP	IC_SDA_SETUP	IC_SDA_SETUP	IC_SDA_SETUP

**Table 15.4:** Derivation of I2C Timing Parameters from \*CNT Registers

### 15.2.15.1 Minimum High and Low Counts in SS, FS, FM+ and HS Modes With IC\_CLK\_FREQ\_OPTIMIZATION = 0.

When the DW\_apb\_i2c operates as an I2C master, in both transmit and receive transfers:

- IC\_SS\_SCL\_LCNT and IC\_FS\_SCL\_LCNT register values must be larger than IC\_FS\_SPKLEN + 7.
- IC\_SS\_SCL\_HCNT and IC\_FS\_SCL\_HCNT register values must be larger than IC\_FS\_SPKLEN + 5.
- If the component is programmed to support HS, IC\_HS\_SCL\_LCNT register value must be larger than IC\_HS\_SPKLEN + 7.
- If the component is programmed to support HS, IC\_HS\_SCL\_HCNT register value must be larger than IC\_HS\_SPKLEN + 5.

Details regarding the DW\_apb\_i2c high and low counts are as follows:

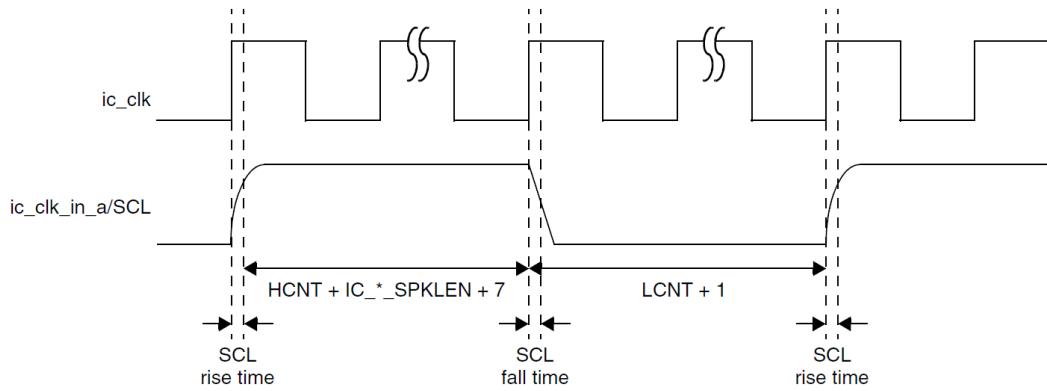
- The minimum value of IC\_\*\_SPKLEN + 7 for the \*\_LCNT registers is due to the time required for the DW\_apb\_i2c to drive SDA after a negative edge of SCL.
- The minimum value of IC\_\*\_SPKLEN + 5 for the \*\_HCNT registers is due to the time required for the DW\_apb\_i2c to sample SDA during the high period of SCL.
- The DW\_apb\_i2c adds one cycle to the programmed \*\_LCNT value in order to generate the low period of the SCL clock; this is due to the counting logic for SCL low counting to (\*\_LCNT + 1).
- The DW\_apb\_i2c adds IC\_\*\_SPKLEN + 7 cycles to the programmed \*\_HCNT value in order to generate the high period of the SCL clock; this is due to the following factors:
  - The counting logic for SCL high counts to (\*\_HCNT+1).
  - The digital filtering applied to the SCL line incurs a delay of SPKLEN + 2 ic\_clk cycles, where SPKLEN is:
    - \* IC\_FS\_SPKLEN if the component is operating in SS or FS
    - \* IC\_HS\_SPKLEN if the component is operating in HS.This filtering includes metastability removal and the programmable spike suppression on SDA and SCL edges.
  - Whenever SCL is driven 1 to 0 by the DW\_apb\_i2c—that is, completing the SCL high time—an internal logic latency of three ic\_clk cycles is incurred. Consequently, the minimum SCL low time of which the DW\_apb\_i2c is capable is nine (9) ic\_clk periods (7 + 1 + 1), while the minimum SCL high time is thirteen (13) ic\_clk periods (6 + 1 + 3 + 3).

#### Note

The total high time and low time of SCL generated by the DW\_apb\_i2c master is also influenced by the rise time and fall time of the SCL line, as shown in the illustration and equations in Figure 15.32 . It should be noted that the SCL rise and fall time parameters vary, depending on external factors such as:

- Characteristics of IO driver
- Pull-up resistor value
- Total capacitance on SCL line, and so on

These characteristics are beyond the control of the DW\_apb\_i2c.



$$\begin{aligned} SCL\_High\_time &= [(HCNT + IC\_SPKLEN + 7) * ic\_clk] + SCL\_Fall\_time \\ SCL\_Low\_time &= [(LCNT + 1) * ic\_clk] - SCL\_Fall\_time + SCL\_Rise\_time \end{aligned}$$

**Figure 15.32:** Impact of SCL Rise Time and Fall Time on Generated SCL

#### 15.2.15.2 Minimum High and Low Counts in SS, FS, FM+ and HS Modes With `IC_CLK_FREQ_OPTIMIZATION = 1`

The minimum high and low counts in SS, FS, FM+ and HS Modes with the `IC_CLK_FREQ_OPTIMIZATION` parameter set to one is such that:

- The total SCL LOW period is driven by `DW_apb_i2c` will be `IC_*_LCNT` register value. The hardware does not support a value less than 6 to be written to the `IC_*_LCNT` register. Additionally, the minimum SCL low time of which the `DW_apb_i2c` is capable is 6 `ic_clk` periods.
- The total SCL HIGH period driven by `DW_apb_i2c` will be `IC_*_HCNT` register value + `SPKLEN` + 3. Additionally, the minimum SCL high time of which the `DW_apb_i2c` is capable is 5 `ic_clk` periods [1+1+3].

The total high time and low time of SCL generated by the `DW_apb_i2c` master is also influenced by the rise time and fall time of the SCL line. The SCL rise and fall time parameters vary depending on external factors such as:

- Characteristics of IO driver
- Pull-up resistor value
- Total capacitance on SCL line, and so on

These characteristics are beyond the control of the `DW_apb_i2c`.

#### 15.2.15.3 Minimum High and Low counts in Ultra-Fast mode (`IC_ULTRA_FAST_MODE = 1`)

When the `DW_apb_i2c` operates as an I2C master:

- The `IC_UFM_SCL_HCNT` register value must be equal or larger than 3.
- The `IC_UFM_SCL_LCNT` register Value must be equal or larger than 5.

#### 15.2.15.4 Minimum IC\_CLK Frequency

This section describes the minimum ic\_clk frequencies that the DW\_apb\_i2c supports for each speed mode, and the associated high and low count values. In Slave mode, IC\_SDA\_HOLD (Thd:dat) and IC\_SDA\_SETUP (Tsu:dat) need to be programmed to satisfy the I2C protocol timing requirements. The following examples are for the case where IC\_FS\_SPKLEN and IC\_HS\_SPKLEN are programmed to 2.

##### 15.2.15.4.1 Standard Mode (SM), Fast Mode (FM), and Fast Mode Plus (FM+) with IC\_CLK\_FREQ\_OPTIMIZATION = 0

This section details how to derive a minimum ic\_clk value for standard and fast modes of the DW\_apb\_i2c. Although the following method shows how to do fast mode calculations, you can also use the same method in order to do calculations for standard mode and fast mode plus.

**Note**

The following computations do not consider the SCL\_Rise\_time and SCL\_Fall\_time.

Given conditions and calculations for the minimum DW\_apb\_i2c ic\_clk value in fast mode:

- Fast mode has data rate of 400kb/s; implies SCL period of 1/400khz = 2.5us
- Minimum hcnt value of 14 as a seed value; IC\_HCNT\_FS = 14
- Protocol minimum SCL high and low times:
  - MIN\_SCL\_LOWtime\_FS = 1300ns
  - MIN\_SCL\_HIGHTime\_FS = 600ns

Derived equations:

$$\frac{SCL\_PERIOD\_FS}{IC\_HCNT\_FS + IC\_LCNT\_FS} = IC\_CLK\_PERIOD$$

$$IC\_LCNT\_FS \times IC\_CLK\_PERIOD = MIN\_SCL\_LOWtime\_FS$$

Combined, the previous equations produce the following:

$$IC\_LCNT\_FS \times \frac{SCL\_PERIOD\_FS}{IC\_LCNT\_FS + IC\_HCNT\_FS} = MIN\_SCL\_LOWtime\_FS$$

Solving for IC\_LCNT\_FS:

$$IC\_LCNT\_FS \times \frac{2.5\mu s}{IC\_LCNT\_FS + 14} = 1.3\mu s$$

The previous equation gives:

$$IC\_LCNT\_FS = roundup(15.166) = 16$$

These calculations produce IC\_LCNT\_FS = 16 and IC\_HCNT\_FS = 14, giving an ic\_clk value of:

$$\frac{2.5\mu s}{16 + 14} = 83.3ns = 12Mhz$$

Testing these results shows that protocol requirements are satisfied.

##### 15.2.15.4.2 High-Speed (HS) Mode With IC\_CLK\_FREQ\_OPTIMIZATION = 0

The method used for standard and fast modes can also be used to derive ic\_clk values for high-speed modes. For

example, given a high-speed mode with a 100pf bus loading, using the standard and fast modes method produces the following:

- IC\_LCNT\_HS = 17
- IC\_HCNT\_HS = 14
- ic\_clk = 105.4 MHz

Table 15.5 lists the minimum ic\_clk values for all modes with high and low count values.

Speed Mode	ic_clkfreq (MHz)	Minimum Value of IC_*_SPKLEN	SCL Low Time in ic_clks	SCL Low Program Value	SCL Low Time	SCL High Time in ic_clks	SCL High Program Value	SCL High Time
SS	2.7	1	13	12	4.7 us	14	6	5.2 us
FS	12	1	16	15	1.33 us	14	6	1.16 us
FM+	32	2	16	15	500 ns	16	7	500 ns
HS (400pf)	51	1	17	16	333 ns	14	6	274 ns
HS (100pf)	105.4	1	17	16	161 ns	14	6	132 ns

**Table 15.5: ic\_clk in Relation to High and Low Counts  
When IC\_CLK\_FREQ\_OPTIMIZATION = 0**

#### Note

- The IC\_\*\_SCL\_LCNT and IC\_\*\_SCL\_HCNT registers are programmed using the SCL low and high program values in Table 15.5, which are calculated using SCL low count minus 1, and SCL high counts minus 8, respectively. The values in Table 15.5 are based on IC\_SDA\_RX\_HOLD = 0. The maximum IC\_SDA\_RX\_HOLD value depends on the IC\_\*CNT registers in Master mode, as described in "SDA Hold Timings in Receiver".
- In order to compute the HCNT and LCNT considering RC timings, use the following equations:  $IC_HCNT_* = [(HCNT + IC_*_SPKLEN + 7) * ic_clk] + SCL_Fall_time$   $IC_LCNT_* = [(LCNT + 1) * ic_clk] - SCL_Fall_time + SCL_Rise_time$

#### 15.2.15.4.3 SM, FM, FM+ and HS Modes With IC\_CLK\_FREQ\_OPTIMIZATION = 1

##### 15.2.15.4.3.1 Master Mode

This section describes the minimum ic\_clk frequencies that the DW\_apb\_i2c supports for each speed mode and the associated high and low count values. The following examples are for the case where IC\_FS\_SPKLEN = 1, IC\_HS\_SPKLEN = 1 and IC\_CLK\_FREQ\_OPTIMIZATION = 1. Below calculations show how to derive a minimum ic\_clk value for fast mode of the DW\_apb\_i2c. Although the following method shows how to do fast mode calculations, you can also use the same method in order to do calculations for any speed mode.

#### Note

The computation in this section does not consider SCL\_Rise\_time and SCL\_Fall\_time.

Following are the conditions and calculations for the minimum DW\_apb\_i2c ic\_clk value in fast mode:

- Fast mode has data rate of 400kb/s; implies SCL period of  $1/400\text{KHz} = 2.5 \text{ us}$
- Minimum hcnt value of 5 as a seed value; IC\_HCNT\_FS = 5
- Protocol minimum SCL high and low times:
  - $\text{MIN\_SCL\_LOWtime\_FS} = 1300 \text{ ns}$
  - $\text{MIN\_SCL\_HIGHtime\_FS} = 600 \text{ ns}$

Following are the derived equations:  $\text{SCL\_PERIOD\_FS}/(\text{IC\_HCNT\_FS} + \text{IC\_LCNT\_FS}) = \text{IC\_CLK\_PERIOD}$   $\text{IC\_LCNT\_FS} \tilde{=} \text{IC\_CLK\_PERIOD} - \text{MIN\_SCL\_LOWtime\_FS}$  Following is the result of combining previous equations:  $\text{IC\_LCNT\_FS} \tilde{=} \text{SCL\_PERIOD\_FS} / (\text{IC\_LCNT\_FS} + \text{IC\_HCNT\_FS}) - \text{MIN\_SCL\_LOWtime\_FS}$  By solving for IC\_LCNT\_FS:  $\text{IC\_LCNT\_FS} \tilde{=} 2.5 (\mu\text{s}) / (\text{IC\_LCNT\_FS} + 5) = 1.3 (\mu\text{s})$  The previous equation provides:  $\text{IC\_LCNT\_FS} = \text{roundup}(5.417) = 6$

#### Note

Minimum IC\_\*\_LCNT value should be equal 6. If derived value is less than 6, consider IC\_LCNT\_FS as 6 only. These calculations produce IC\_LCNT\_FS = 6 and IC\_HCNT\_FS = 5, providing an ic\_clk value of:  $2.5 (\mu\text{s}) / (6 + 5) = 227.27\text{ns} = 4.4 \text{ MHz}$  Testing these results shows that the protocol requirements are satisfied. Table 15.6 lists the minimum ic\_clk values for all modes with high and low count values.

Speed Mode	ic_clk Frequency (MHz)	Minimum Value of IC_*_SPKLEN	SCL Low Time in ic_clks	SCL Low Program Value	SCL Low Time in ns	SCL High Time in ic_clks	SCL High Program Value	SCL High Time in ns
SS	1.1	1	6	6	5454. 545	5	1	455. 455
FS	4.4	1	6	6	1363. 636	5	1	1136. 364
FM+	11	1	6	6	545. 4545	5	1	454. 5455
HS (400pf)	18.7	1	6	6	320. 8527	5	1	267. 3773
HS (100pf)	37.4	1	6	6	160. 4236	5	1	133. 6864

**Table 15.6:** ic\_clk in Relation to High and Low Counts  
When IC\_CLK\_FREQ\_OPTIMIZATION = 1

#### Note

- The IC\_\*\_SCL\_LCNT and IC\_\*\_SCL\_HCNT registers are programmed using the SCL low and high program values in Table 15.6, which are calculated as SCL low count, and SCL high count minus 4, respectively. The values in Table 15.6 are based on IC\_SDA\_RX\_HOLD = 0. The maximum IC\_SDA\_RX\_HOLD value depends on the IC\_\*\_CNT registers in master mode, as described in "SDA Hold Timings in Receiver".
- To compute the HCNT and LCNT considering RC timings, use the following equations:  $\text{IC\_HCNT\_*} = [(\text{HCNT} + \text{IC\_*_SPKLEN} + 3) * \text{ic\_clk}] + \text{SCL\_Fall\_time}$   $\text{IC\_LCNT\_*} = [\text{LCNT} * \text{ic\_clk}] - \text{SCL\_Fall\_time} + \text{SCL\_Rise\_time}$

### 15.2.15.4.3.2 Slave Mode

DW\_apb\_i2c in slave mode requires minimum 5 ic\_clk cycles [SPKLEN + 3 (Metastability removal, worst case) + 1] to drive SDA after a falling edge of SCL. Therefore, the ic\_clk frequency must be selected such that the maximum data hold time (thd;dat)/data valid time (tVD;DAT) is not violated. For example, in high-speed mode with a 100pf bus loading (SCLH clock frequency upto 3.4 MHz), the maximum data hold time is 70 ns. Therefore, the minimum frequency in which DW\_apb\_i2c can operate in slave mode without violating thd;dat is  $70\text{ns}/5 = 14\text{ns} = 71.42\text{ MHz}$ . Table 15.7 lists the minimum IC\_CLK frequency in slave mode when IC\_CLK\_FREQ\_OPTIMIZATION is set to 1.

Speed Mode	ic_clk Frequency (MHz)	Minimum Value of IC_*_SPKLEN	Minimum data hold time in ic_clks	Maximum data hold time
SS	1.45	1	5	3.45 us
FS	5.56	1	5	0.9 us
FM+	11.11	1	5	0.45 us
HS (400pf)	35.71	1	5	140 ns
HS (100pf)	71.42	1	5	70 ns

**Table 15.7:** Minimum IC\_CLK Frequency in Slave Mode  
When IC\_CLK\_FREQ\_OPTIMIZATION=1

### 15.2.15.4.4 ULTRA-FAST Mode

#### 15.2.15.4.4.1 Master mode

This section describes the minimum ic\_clk frequency that the DW\_apb\_i2c supports for Ultra-Fast speed mode and the associated high and low count values. The following calculations show how to derive a minimum ic\_clk value.

##### Note

The following computations do not consider the SCL\_Rise\_time and SCL\_Fall\_time.

Given conditions and calculations for the minimum DW\_apb\_i2c ic\_clk value in Ultra-Fast mode:

- Fast mode has data rate of 5000kb/s; implies SCL period of  $1/5000\text{kHz} = 200\text{ns}$
- Minimum hcnt value of 3 as a seed value; IC\_UFM\_SCL\_HCNT = 3
- Protocol minimum SCL high and low times:
  - MIN\_SCL\_LOWtime\_UFm = 50 ns
  - MIN\_SCL\_HIGHtime\_UFm = 50ns

Derived equations:

- $\text{SCL\_PERIOD\_UFm}/(\text{IC\_HCNT\_UFm} + \text{IC\_LCNT\_UFm}) = \text{IC\_CLK\_PERIOD}$
- $\text{IC\_LCNT\_UFm} \cdot \text{IC\_CLK\_PERIOD} = \text{MIN\_SCL\_LOWtime\_UFm}$

Combined, the previous equations produce the following:  $\text{IC\_LCNT\_UFm} \cdot \text{IC\_CLK\_PERIOD} / (\text{IC\_LCNT\_UFm} + \text{IC\_HCNT\_UFm}) = \text{MIN\_SCL\_LOWtime\_UFm}$  Solving for IC\_LCNT\_UFm:  $\text{IC\_LCNT\_UFm} \cdot 200\text{ns} / (\text{IC\_LCNT\_UFm} + 3) = 50\text{ns}$  The previous equation gives:  $\text{IC\_LCNT\_UFm} = 1$

**Note**

Minimum IC\_SCL\_UFM\_LCNT value should be equal 5. If derived value is less than 5, consider IC\_LCNT\_UFm as 5 only.

These calculations produce IC\_LCNT\_UFm = 5 and IC\_HCNT\_UFm = 3, giving an ic\_clk value of:  $200 \text{ ns}/(5 + 3) = 25\text{ns} = 40\text{Mhz}$  Testing these results shows that protocol requirements are satisfied. Table 15.8 describes the relation between the High and Low counts with ic\_clk frequency

Speed	ic_clk (freq) (Mhz)	SCL Low Program Value	SCL Low Time in ic_clks	SCL Low Time	SCL High Program Value	SCL HighT- ime in ic_clks	SCL HighT- ime
UltraFast Mode	40	5	5	125 ns	3	3	75 ns

**Table 15.8:** ic\_clk in relation to High and Low Counts  
when IC\_ULTRA\_FAST\_MODE=1

**Note**

- The IC\_UFM\_SCL\_LCNT and IC\_UFM\_SCL\_HCNT registers are programmed using the SCL low and high program values in Table 15.8, which are calculated as SCL low count, and SCL high count, respectively. The values in Table 15.8 are based on IC\_SDA\_RX\_HOLD = 0. The maximum IC\_SDA\_RX\_HOLD value depends on the IC\_UFM\_SCL\_LCNT registers in Master mode, as described in "SDA Hold Timings in Receiver".
- In order to compute the HCNT and LCNT considering RC timings, use the following equations:  $\text{IC_UFM_SCL_HCNT} = [\text{HCNT} * \text{ic_clk}] + \text{SCL_Fall_time}$   $\text{IC_UFM_SCL_LCNT} = [\text{LCNT} * \text{ic_clk}] - \text{SCL_Fall_time} + \text{SCL_Rise_time}$

#### 15.2.15.4.4.2 Slave mode

DW\_apb\_i2c in slave mode requires minimum of 2 ic\_clk cycles for SCL High period and SCL Low Period. Therefore, the minimum ic\_clk frequency for the slave mode is 40 MHz.

#### 15.2.15.4.5 Calculating High and Low Counts with IC\_CLK\_FREQ\_OPTIMIZATION = 0

The calculations below show how to calculate SCL high and low counts for each speed mode in the DW\_apb\_i2c. For the calculations to work, the ic\_clk frequencies used must not be less than the minimum ic\_clk frequencies specified in Table 15.5. The DW\_apb\_i2c coreConsultant GUI can automatically calculate SCL high and low count values. By specifying an integer ic\_clk period value in nanoseconds for the IC\_CLK\_PERIOD parameter, SCL high and low count values are automatically calculated for each speed mode. The ic\_clk period must not specify a clock of a lower frequency than required for all supported speed modes. It is possible that the automatically calculated values may result in a baud rate higher than the maximum rate specified by the protocol. If this happens, either the low or high count values can be scaled up to reduce the baud rate. The equation to calculate the proper number of ic\_clk signals required for setting the proper SCL clocks high and low times is as follows:

$\text{IC_xCNT} = (\text{ROUNDUP}(\text{MIN_SCL_xxtime} * \text{OSCFREQ}, 0))$  ROUNDUP is an explicit Excel function call that is used to convert a real number to its equivalent integer number.  $\text{MIN_SCL_HIGHTime}$  = Minimum High Period  $\text{MIN_SCL_HIGHTime} = 4000 \text{ ns}$  for 100 kbps  $600 \text{ ns}$  for 400 kbps  $260 \text{ ns}$  for 1000 kbps  $60 \text{ ns}$  for 3.4 Mbs, bus loading =  $100\text{pF}$   $120 \text{ ns}$  for 3.4 Mbs, bus loading =  $400\text{pF}$   $\text{MIN_SCL_LOWtime}$  = Minimum Low Period  $\text{MIN_SCL_LOWtime} = 4700 \text{ ns}$  for 100 kbps  $1300 \text{ ns}$  for 400 kbps  $500 \text{ ns}$  for 1000 kbps  $160 \text{ ns}$  for 3.4Mbs, bus loading =  $100\text{pF}$   $320 \text{ ns}$  for 3.4Mbs, bus loading =  $400\text{pF}$  OSCFREQ = ic\_clk Clock Frequency (Hz). For example: OSCFREQ = 100 MHz I2Cmode = fast, 400 kbit/s  $\text{MIN_SCL_HIGHTime} = 600 \text{ ns}$ .  $\text{MIN_SCL_LOWtime} = 1300 \text{ ns}$ .  $\text{IC_xCNT} = (\text{ROUNDUP}(\text{MIN_SCL_HIGH\_LOWtime} * \text{OSCFREQ}, 0))$   $\text{IC_HCNT} = (\text{ROUNDUP}(600 \text{ ns} * 100 \text{ MHz}, 0))$   $\text{IC_HCNTSCL}$

PERIOD = 60 IC\_LCNT = (ROUNDUP(1300 ns \* 100 MHz,0)) IC\_LCNTSCL PERIOD = 130 Actual MIN\_SCL\_HIGHTime = 60\*(1/100 MHz) = 600 ns Actual MIN\_SCL\_LOWtime = 130\*(1/100 MHz) = 1300 ns

#### Note

Once the default values for SCL HighCount and LowCount are computed by the coreConsultant GUI, check that the values are consistent with the required baud rate. In case the computed values do not match with the required values, you can manually scale the values, as described in the section "High-Speed (HS) Mode With IC\_CLK\_FREQ\_OPTIMIZATION = 0".

#### 15.2.15.4.6 Calculating High and Low counts with IC\_CLK\_FREQ\_OPTIMIZATION = 1

The calculations below show how to calculate SCL high and low counts for each speed mode in the DW\_apb\_i2c. For the calculations to work, the ic\_clk frequencies used must not be less than the minimum ic\_clk frequencies specified in Table 15.6. The DW\_apb\_i2c coreConsultant GUI can automatically calculate SCL high and low count values. By specifying an integer ic\_clk period value in nanoseconds for the IC\_CLK\_PERIOD parameter, SCL high and low count values are automatically calculated for each speed mode. The ic\_clk period must not specify a clock of a lower frequency than required for all supported speed modes. It is possible that the automatically calculated values may result in a baud rate higher than the maximum rate specified by the protocol. If this happens, either the low or high count values can be scaled up to reduce the baud rate. For more information, see "Master Mode". The equation to calculate the proper number of ic\_clk signals required for setting the proper SCL clocks high and low times is as follows:

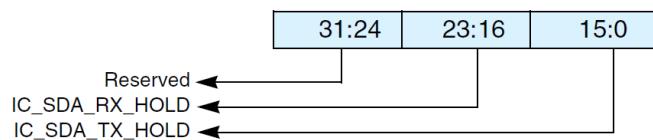
IC\_xCNT = (ROUNDUP(MIN\_SCL\_xxxtimes\*OSCFREQ,0)) ROUNDUP is an explicit Excel function call that is used to convert a real number to its equivalent integer number. MIN\_SCL\_HIGHTime = Minimum High Period MIN\_SCL\_HIGHTime = 4000 ns for 100 kbps 600 ns for 400 kbps 260 ns for 1000 kbps 60 ns for 3.4 Mbps, bus loading = 100pF 160 ns for 3.4 Mbps, bus loading = 400pF MIN\_SCL\_LOWtime = Minimum Low Period MIN\_SCL\_LOWtime = 4700 ns for 100 kbps 1300 ns for 400 kbps 500 ns for 1000 kbps 120 ns for 3.4Mbps, bus loading = 100pF 320 ns for 3.4Mbps, bus loading = 400pF OSCFREQ = ic\_clk Clock Frequency (Hz). For example: OSCFREQ = 100 MHz I2Cmode = fast, 400 kbit/s MIN\_SCL\_HIGHTime = 600 ns. MIN\_SCL\_LOWtime = 1300 ns. IC\_xCNT = (ROUNDUP(MIN\_SCL\_HIGH\_LOWtime\*OSCFREQ,0)) IC\_HCNT = (ROUNDUP(600 ns \* 100 MHz,0)) IC\_HCNTSCL PERIOD = 60 IC\_LCNT = (ROUNDUP(1300 ns \* 100 MHz,0)) IC\_LCNTSCL PERIOD = 130 Actual MIN\_SCL\_HIGHTime = 60\*(1/100 MHz) = 600 ns Actual MIN\_SCL\_LOWtime = 130\*(1/100 MHz) = 1300 ns

#### Note

When the default values for SCL HighCount and LowCount are computed by the coreConsultant GUI, check that the values are consistent with the required baud rate. In case the computed values do not match with the required values, you can manually scale the values, as described in "Master mode".

### 15.2.16 SDA Hold Time

The I2C protocol specification requires 300ns of hold time on the SDA signal ( $t_{HD;DAT}$ ) in standard mode and fast mode, and a hold time long enough to bridge the undefined part between logic 1 and logic 0 of the falling edge of SCL in high speed mode and fast mode plus. Board delays on the SCL and SDA signals can mean that the hold-time requirement is met at the I2C master, but not at the I2C slave (or vice-versa). As each application encounters differing board delays, the DW\_apb\_i2c contains a software programmable register (IC\_SDA\_HOLD) to enable dynamic adjustment of the SDA hold-time. The bits [15:0] are used to control the hold time of SDA during transmit in both slave and master mode (after SCL goes from HIGH to LOW). The bits [23:16] are used to extend the SDA transition (if any) whenever SCL is HIGH in the receiver (in either master or slave mode).



**Figure 15.33:** IC\_SDA\_HOLD Register

If different SDA hold times are required for different speed modes, the IC\_SDA\_HOLD register must be reprogrammed when the speed mode is being changed. The IC\_SDA\_HOLD register can be programmed only when the DW\_apb\_i2c is disabled (IC\_ENABLE[0] = 0). The reset value of the IC\_SDA\_HOLD register can be set via the coreConsultant parameter IC\_DEFAULT\_SDA\_HOLD

#### 15.2.16.1 SDA Hold Timings in Receiver

When DW\_apb\_i2c acts as a receiver, according to the I2C protocol, the device should internally hold the SDA line to bridge undefined gap between logic 1 and logic 0 of SCL. IC\_SDA\_RX\_HOLD can be used to alter the internal hold time which DW\_apb\_i2c applies to the incoming SDA line. Each value in the IC\_SDA\_RX\_HOLD register represents a unit of one ic\_clk period. The minimum value of IC\_SDA\_RX\_HOLD is 0. This hold time is applicable only when SCL is HIGH. The receiver does not extend the SDA after SCL goes LOW internally. Figure 15.34 shows the DW\_apb\_i2c as receiver with IC\_SDA\_RX\_HOLD programmed to greater than or equal to 3.

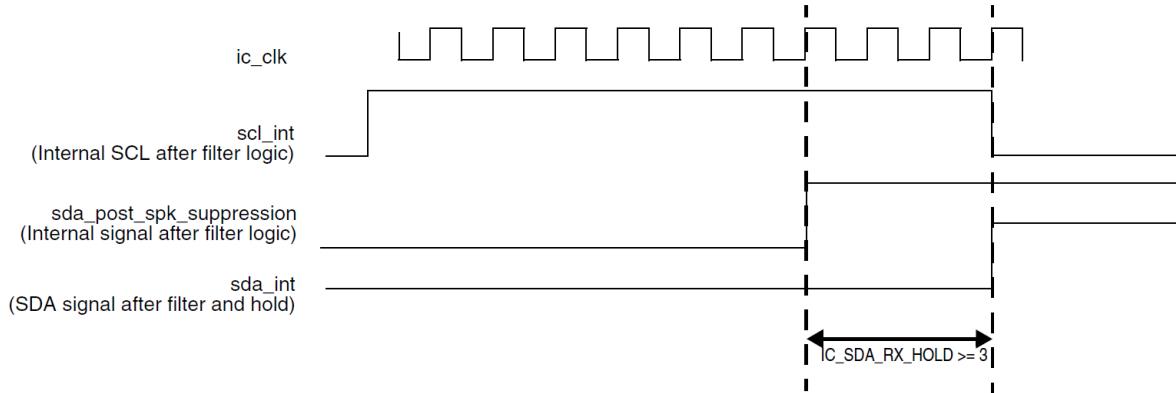


Figure 15.34

If `IC_SDA_RX_HOLD` is greater than 3, `DW_apb_i2c` does not hold SDA beyond 3 `ic_clk` cycles, because SCL goes LOW internally. Figure 15.35 shows the `DW_apb_i2c` as receiver with `IC_SDA_RX_HOLD` programmed to 2.

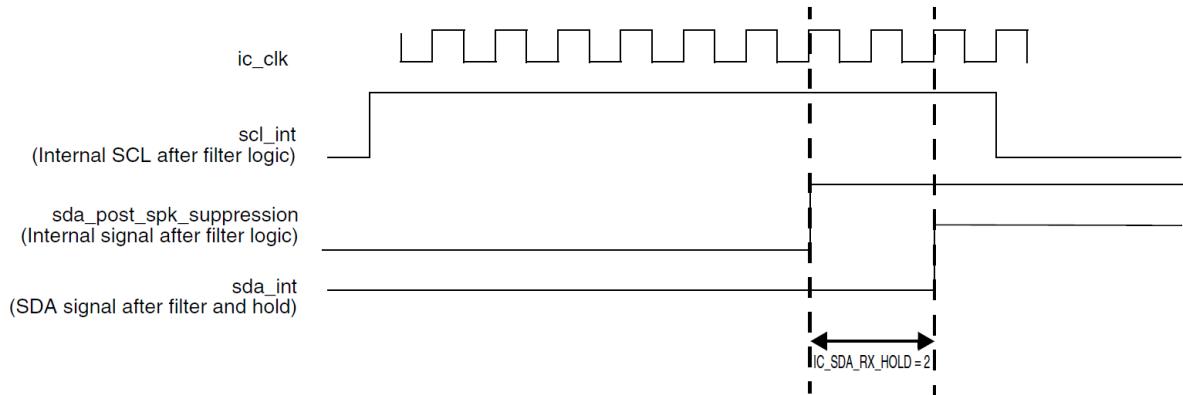


Figure 15.35

The maximum values of `IC_SDA_RX_HOLD` that can be programmed in the register for the respective speed modes are derived from the equations show in Table 15.9.

Speed Mode	Maximum <code>IC_SDA_RX_HOLD</code> Value
Standard Mode	<code>IC_SS_SCL_HCNT - IC_FS_SPKLEN - 3</code>
Fast Mode or Fast Mode Plus	<code>IC_FS_SCL_HCNT - IC_FS_SPKLEN - 3</code>
High Speed ( <code>IC_CAP_LOADING = 100</code> )	<code>Min IC_FS_SCL_HCNT - IC_FS_SPKLEN - 3 , IC_HS_SCL_LCNT - IC_HS_SPKLEN - 3</code>

Table 15.9: Maximum Values for `IC_SDA_RX_HOLD` (Continued on next page)

		continued from previous page
High Speed (IC_CAP_LOADING =400)		Min IC_FS_SCL_HCNT - IC_FS_SPKLEN - 3 , (IC_HS_SCL_LC- NT/2) - IC_HS_SPKLEN - 3

**Table 15.9:** Maximum Values for IC\_SDA\_RX\_HOLD**Note**

The maximum values in Table 15.9 is applicable in Master mode. In Slave mode, make sure the IC\_SDA\_RX\_HOLD does not exceed the maximum SCL fall time (tf in SS and FS mode or tfcl in HS Mode).

**15.2.16.2 SDA Hold Timings in Transmitter**

The IC\_SDA\_TX\_HOLD register can be used to alter the timing of the generated SDA (ic\_data\_oe) signal by the DW\_apb\_i2c. Each value in the IC\_SDA\_TX\_HOLD register represents a unit of one ic\_clk period. When the DW\_apb\_i2c is operating in Master Mode, the minimum tHD:DAT timing is one ic\_clk period. Therefore even when IC\_SDA\_TX\_HOLD has a value of zero, the DW\_apb\_i2c will drive SDA (ic\_data\_oe) one ic\_clk cycle after driving SCL (ic\_clk\_oe) to logic 0. For all other values of IC\_SDA\_TX\_HOLD, the following is true:

- Drive on SDA (ic\_data\_oe) occurs IC\_SDA\_TX\_HOLD ic\_clk cycles after driving SCL (ic\_clk\_oe) to logic 0

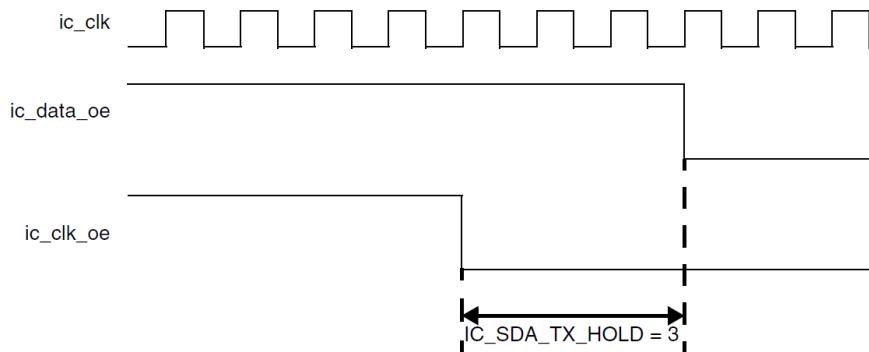
When the DW\_apb\_i2c is operating in Slave Mode, the minimum tHD:DAT timing is SPKLEN + 7 ic\_clk periods, where SPKLEN is:

- IC\_FS\_SPKLEN if the component is operating in standard mode, fast mode, or fast mode plus
- IC\_HS\_SPKLEN if the component is operating in high speed mode

This delay allows for synchronization and spike suppression on the SCL (ic\_clk\_in\_a) sample. Therefore, even when IC\_SDA\_TX\_HOLD has a value less than SPKLEN + 7, the DW\_apb\_i2c drives SDA (ic\_data\_oe) SPKLEN + 7 ic\_clk cycles after SCL (ic\_clk\_in) has transitioned to logic 0. For all other values of IC\_SDA\_TX\_HOLD, the following is true:

- Drive on SDA (ic\_data\_oe) occurs IC\_SDA\_TX\_HOLD ic\_clk cycles after SCL (ic\_clk\_in\_a) has transitioned to logic 0.

Figure 15.36 shows the tHD:DAT timing generated by the DW\_apb\_i2c operating in Master Mode when IC\_SDA\_TX\_HOLD = 3.



**Figure 15.36:** DW\_apb\_i2c Master Implementing thD;DAT with IC\_SDA\_HOLD = 3

#### Note

The programmed SDA hold time cannot exceed at any time the duration of the low part of scl. Therefore the programmed value cannot be larger than N\_SCL\_LOW-2, where N\_SCL\_LOW is the duration of the low part of the scl period measured in ic\_clk cycles.

### 15.2.17 DMA Controller Interface

The DW\_apb\_i2c has an optional built-in DMA capability that can be selected at configuration time; it has a hand-shaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA. While the DW\_apb\_i2c DMA operation is designed in a generic way to fit any DMA controller as easily as possible, it is designed to work seamlessly, and best used, with the DesignWare DMA Controller, the DW\_ahb\_dmac. The settings of the DW\_ahb\_dmac that are relevant to the operation of the DW\_apb\_i2c are discussed here, mainly bit fields in the DW\_ahb\_dmac channel control register, CTLx, where x is the channel number.

#### Note

When the DW\_apb\_i2c interfaces to the DW\_ahb\_dmac, the DW\_ahb\_dmac is always a flow controller; that is, it controls the block size. This must be programmed by software in the DW\_ahb\_dmac. The DW\_ahb\_dmac always transfers data using DMA burst transactions if possible, for efficiency. For more information, refer to the DesignWare DW\_ahb\_dmac Databook. Other DMA controllers act in a similar manner.

The relevant DMA settings are discussed in the following sections.

#### Note

The DMA output **dma\_finish** is a status signal to indicate that the DMA block transfer is complete. DW\_apb\_i2c does not use this status signal, and therefore does not appear in the I/O port list.

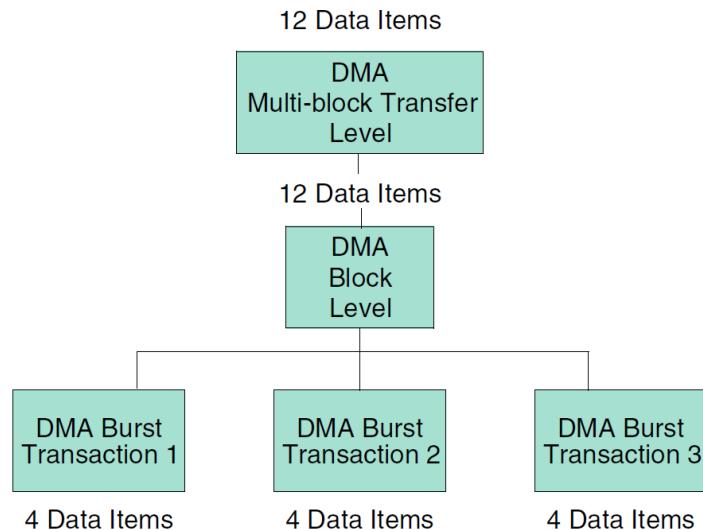
#### 15.2.17.1 Enabling the DMA Controller Interface

To enable the DMA Controller interface on the DW\_apb\_i2c, you must write the DMA Control Register (IC\_DMA\_CR). Writing a 1 into the TDMAE bit field of IC\_DMA\_CR register enables the DW\_apb\_i2c transmit handshaking inter-

face. Writing a 1 into the RDMAE bit field of the IC\_DMA\_CR register enables the DW\_apb\_i2c receive handshaking interface.

### 15.2.17.2 Overview of Operation

As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by DW\_apb\_i2c; this is programmed into the BLOCK\_TS field of the DW\_ahb\_dmac CTLx register. The block is broken into a number of transactions, each initiated by a request from the DW\_apb\_i2c. The DMA Controller must also be programmed with the number of data items (in this case, DW\_apb\_i2c FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length and is programmed into the SRC\_MSIZE/DEST\_MSIZE fields of the DW\_ahb\_dmac CTLx register for source and destination, respectively. Figure 15.37 shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4. In this case, the block size is a multiple of the burst transaction length. Therefore, the DMA block transfer consists of a series of burst transactions. If the DW\_apb\_i2c makes a transmit request to this channel, four data items are written to the DW\_apb\_i2c TX FIFO. Similarly, if the DW\_apb\_i2c makes a receive request to this channel, four data items are read from the DW\_apb\_i2c RX FIFO. Three separate requests must be made to this DMA channel before all 12 data items are written or read.



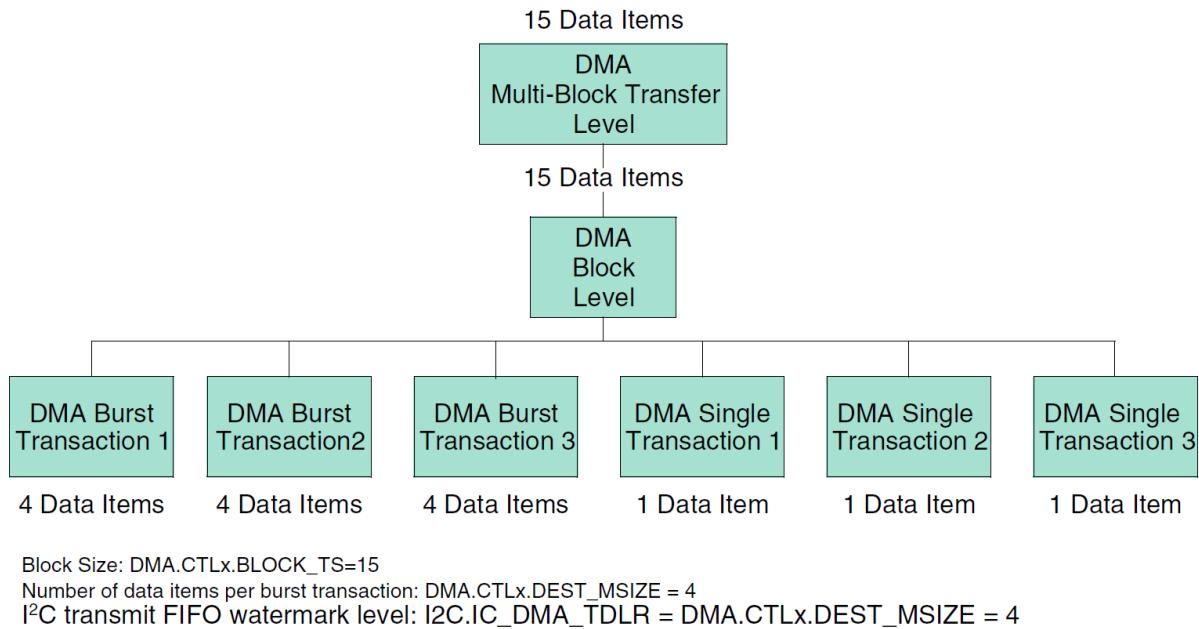
Block Size: DMA.CTLx.BLOCK\_TS=12

Number of data items per source burst transaction: DMA.CTLx.SRC\_MSIZE = 4

I<sup>2</sup>C receive FIFO watermark level: I2C.DMARDLR + 1 = DMA.CTLx.SRC\_MSIZE = 4

**Figure 15.37:** Breakdown of DMA Transfer into Burst Transactions

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in Figure 15.38, a series of burst transactions followed by single transactions are needed to complete the block transfer.



**Figure 15.38:** Breakdown of DMA Transfer into Single and Burst Transactions

### 15.2.17.3 Transmit Watermark Level and Transmit FIFO Underflow

During DW\_apb\_i2c serial transfers, transmit FIFO requests are made to the DW\_ahb\_dmac whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (IC\_DMA\_TDRL) value; this is known as the watermark level. The DW\_ahb\_dmac responds by writing a burst of data to the transmit FIFO buffer, of length CTLx.DEST\_MSIZE. If IC\_EMPTYFIFO\_HOLD\_MASTER\_EN parameter is set to 0, data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise, the FIFO will run out of data causing a STOP to be inserted on the I2C bus. To prevent this condition, the user must set the watermark level correctly.

### 15.2.17.4 Choosing the Transmit Watermark Level

Consider the example where the assumption is made: DMA.CTLx.DEST\_MSIZE = FIFO\_DEPTH - I2C.IC\_DMA\_TDRL. Here the number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.

#### 15.2.17.4.1 Case 1: IC\_DMA\_TDRL = 2

- Transmit FIFO watermark level = I2C.IC\_DMA\_TDRL = 2
- DMA.CTLx.DEST\_MSIZE = FIFO\_DEPTH - I2C.IC\_DMA\_TDRL = 6
- I2C transmit FIFO\_DEPTH = 8
- DMA.CTLx.BLOCK\_TS = 30

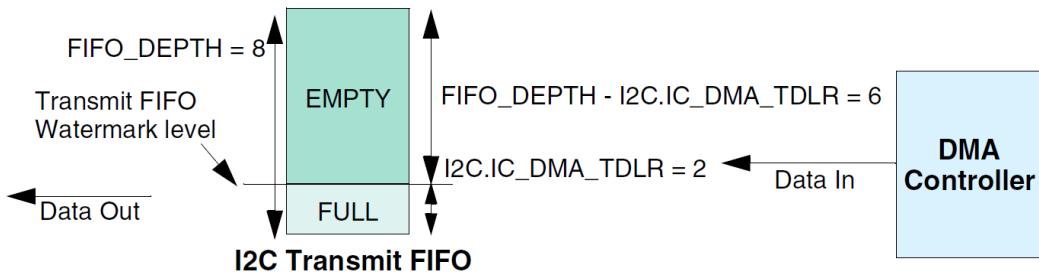


Figure 15.39: Case 1 Watermark Levels

Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:  $\text{DMA.CTLx.BLOCK_TS}/\text{DMA.CTLx.DEST_MSIZE} = 30/6 = 5$ . The number of burst transactions in the DMA block transfer is 5. But the watermark level,  $\text{I2C.IC_DMA_TDRL}$ , is quite low. Therefore, the probability of an I2C underflow is high where the I2C serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.

#### 15.2.17.4.2 Case 2: IC\_DMA\_TDRL = 6

- Transmit FIFO watermark level =  $\text{I2C.IC_DMA_TDRL} = 6$
- $\text{DMA.CTLx.DEST_MSIZE} = \text{FIFO_DEPTH} - \text{I2C.IC_DMA_TDRL} = 2$
- $\text{I2C transmit FIFO\_DEPTH} = 8$
- $\text{DMA.CTLx.BLOCK_TS} = 30$

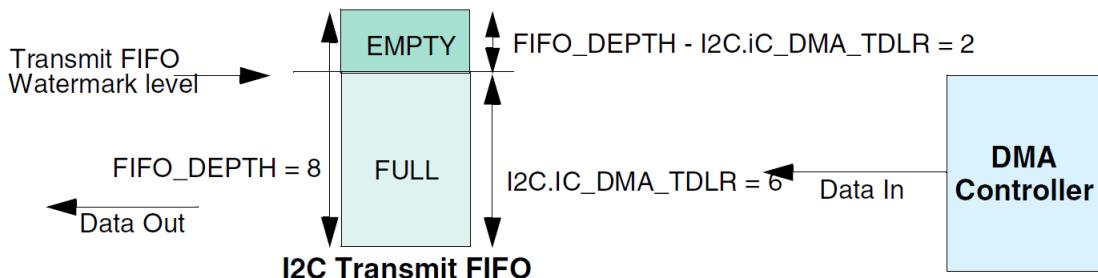


Figure 15.40: Case 2 Watermark Levels

Number of burst transactions in Block:  $\text{DMA.CTLx.BLOCK_TS}/\text{DMA.CTLx.DEST_MSIZE} = 30/2 = 15$

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level,  $\text{I2C.IC_DMA_TDRL}$ , is high. Therefore, the probability of an I2C underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the I2C transmit FIFO becomes empty. Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block.

This provides a potentially greater amount of AMBA bursts per block and worse bus utilization than the former case. Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the I2C transmits data to the rate at which the DMA can respond to destination burst requests. For example, promoting the channel to the highest priority channel in the DMA, and promoting the DMA master interface to the highest priority master in the AMBA layer, increases the rate at which the DMA controller can respond to burst transaction requests. This in turn allows the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

#### 15.2.17.5 Selecting DEST\_MSIZEx and Transmit FIFO Overflow

As can be seen from Figure 15.40 , programming DMA.CTLx.DEST\_MSIZEx to a value greater than the watermark level that triggers the DMA request may cause overflow when there is not enough space in the I2C transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow: DMA.CTLx.DEST\_MSIZEx <= I2C.FIFO\_DEPTH - I2C.IC\_DMA\_TDRL (1) In Case 2: IC\_DMA\_TDRL = 6, the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST\_MSIZEx. Thus, the transmit FIFO may be full, but not overflowed, at the completion of the burst transaction. Therefore, for optimal operation, DMA.CTLx.DEST\_MSIZEx should be set at the FIFO level that triggers a transmit DMA request; that is: DMA.CTLx.DEST\_MSIZEx = I2C.FIFO\_DEPTH - I2C.IC\_DMA\_TDRL (2) This is the setting used in Figure 15.38 . Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, and this in turn improves AMBA bus utilization.

##### Note

The transmit FIFO will not be full at the end of a DMA burst transfer if the I2C has successfully transmitted one data item or more on the I2C serial transmit line during the transfer.

#### 15.2.17.6 Receive Watermark Level and Receive FIFO Overflow

During DW\_apb\_i2c serial transfers, receive FIFO requests are made to the DW\_ahb\_dmac whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register; that is, IC\_DMA\_RDLR+1. This is known as the watermark level. The DW\_ahb\_dmac responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC\_MSIZEx. Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO will fill with data (overflow). To prevent this condition, the user must correctly set the watermark level.

#### 15.2.17.7 Choosing the Receive Watermark level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, IC\_DMA\_RDLR+1, should be set to minimize the probability of overflow, as shown in Figure 15.41. It is a trade-off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

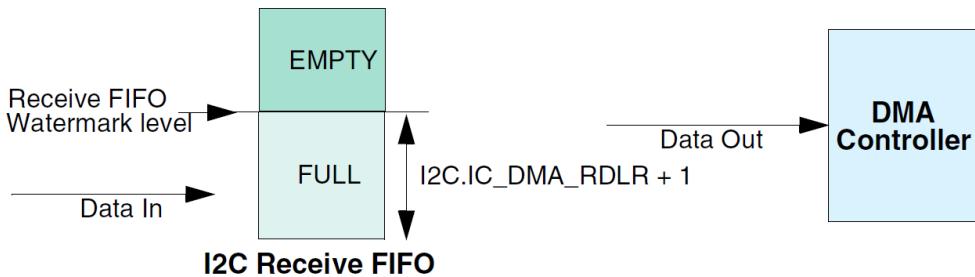
#### 15.2.17.8 Selecting SRC\_MSIZEx and Receive FIFO Underflow

As can be seen in Figure 15.41, programming a source burst transaction length greater than the watermark level may cause underflow when there is not enough data to service the source burst request. Therefore, equation 3 below must be adhered to avoid underflow. If the number of data items in the receive FIFO is equal to the source

burst length at the time the burst request is made - DMA.CTLx.SRC\_MSIZEx - the receive FIFO may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC\_MSIZEx should be set at the watermark level; that is: DMA.CTLx.SRC\_MSIZEx = I2C.IC\_DMA\_RDLR + 1 (3) Adhering to equation (3) reduces the number of DMA bursts in a block transfer, which in turn can avoid underflow and improve AMBA bus utilization.

#### Note

The receive FIFO will not be empty at the end of the source burst transaction if the I2C has successfully received one data item or more on the I2C serial receive line during the burst.



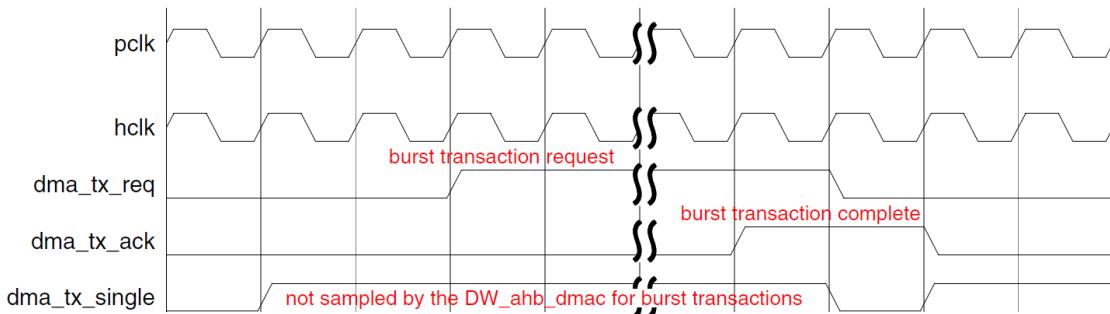
**Figure 15.41:** I2C Receive FIFO

### 15.2.17.9 Handshaking Interface Operation

The following sections discuss the handshaking interface.

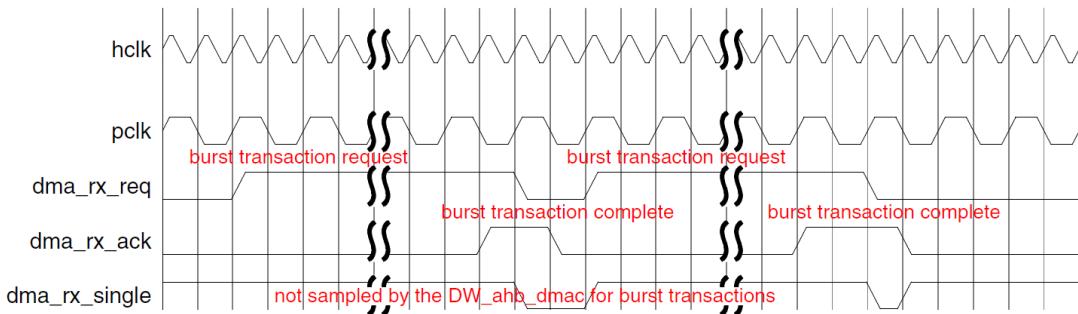
#### 15.2.17.9.1 dma\_tx\_req, dma\_rx\_req

The request signals for source and destination, dma\_tx\_req and dma\_rx\_req, are activated when their corresponding FIFOs reach the watermark levels as discussed earlier. The DW\_ahb\_dmac uses rising-edge detection of the dma\_tx\_req/dma\_rx\_req to identify a request on the channel. Upon reception of the dma\_tx\_ack/dma\_rx\_ack signal from the DW\_ahb\_dmac to indicate the burst transaction is complete, the DW\_apb\_i2c de-asserts the burst request signals, dma\_tx\_req/dma\_rx\_req, until dma\_tx\_ack/dma\_rx\_ack is de-asserted by the DW\_ahb\_dmac. When the DW\_apb\_i2c samples that dma\_tx\_ack/dma\_rx\_ack is de-asserted, it can re-assert the dma\_tx\_req/dma\_rx\_req of the request line if their corresponding FIFOs exceed their watermark levels (back-to-back burst transaction). If this is not the case, the DMA request lines remain de-asserted. Figure 15.42 shows a timing diagram of a burst transaction where pclk = hclk.



**Figure 15.42:** Burst Transaction - pclk = hclk

Figure 15.43 shows two back-to-back burst transactions where the hclk frequency is twice the pclk frequency.



**Figure 15.43:** Back-to-Back Burst Transactions - hclk = 2\*pclk

The handshaking loop is as follows: `dma_tx_req/dma_rx_req` asserted by `DW_apb_i2c` -> `dma_tx_ack/dma_rx_ack` asserted by `DW_ahb_dmac` -> `dma_tx_req/dma_rx_req` de-asserted by `DW_apb_i2c` -> `dma_tx_ack/dma_rx_ack` de-asserted by `DW_ahb_dmac` -> `dma_tx_req/dma_rx_req` reasserted by `DW_apb_i2c`, if back-to-back transaction is required

#### Note

The burst transaction request signals, `dma_tx_req` and `dma_rx_req`, are generated in the `DW_apb_i2c` off `pclk` and sampled in the `DW_ahb_dmac` by `hclk`. The acknowledge signals, `dma_tx_ack` and `dma_rx_ack`, are generated in the `DW_ahb_dmac` off `hclk` and sampled in the `DW_apb_i2c` of `pclk`. The handshaking mechanism between the `DW_ahb_dmac` and the `DW_apb_i2c` supports quasi-synchronous clocks; that is, `hclk` and `pclk` must be phase-aligned, and the `hclk` frequency must be a multiple of the `pclk` frequency.

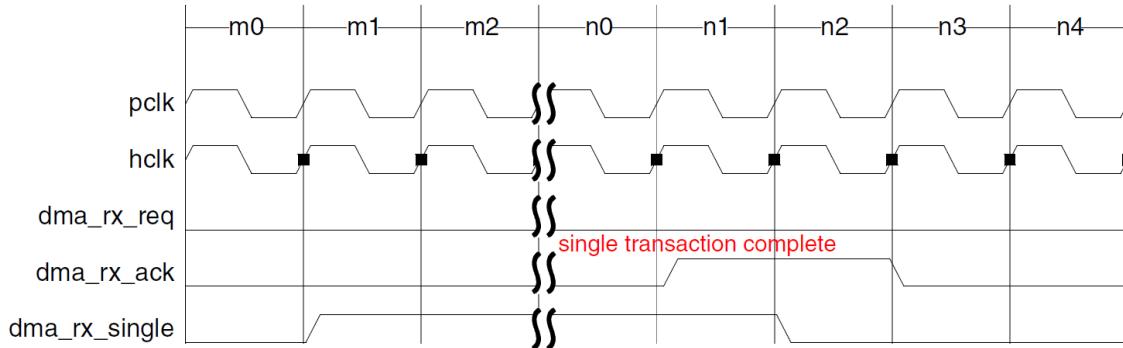
Two things to Note here:

1. The burst request lines, `dma_tx_req` signal/`dma_rx_req`, once asserted remain asserted until their corresponding `dma_tx_ack`/`dma_rx_ack` signal is received even if the respective FIFO's drop below their watermark levels during the burst transaction.
2. The `dma_tx_req`/`dma_rx_req` signals are de-asserted when their corresponding `dma_tx_ack`/`dma_rx_ack` signals are asserted, even if the respective FIFOs exceed their watermark levels.

### 15.2.17.9.2 dma\_tx\_single, dma\_rx\_single

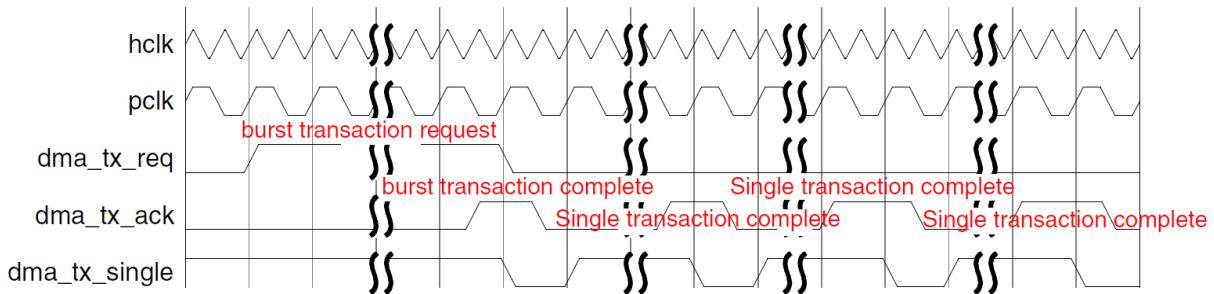
The `dma_tx_single` signal is a status signal. It is asserted when there is at least one free entry in the transmit FIFO and cleared when the transmit FIFO is full. The `dma_rx_single` signal is a status signal. It is asserted when there is at least one valid data entry in the receive FIFO and cleared when the receive FIFO is empty. These signals are needed by only the `DW_ahb_dmac` for the case where the block size, `CTLx.BLOCK_TS`, that is programmed into the `DW_ahb_dmac` is not a multiple of the burst transaction length, `CTLx.SRC_MSIZE`, `CTLx.DEST_MSIZE`, as shown in Figure 15.38 . In this case, the DMA single outputs inform the `DW_ahb_dmac` that it is still possible to perform single data item transfers, so it can access all data items in the transmit/receive FIFO and complete the DMA block transfer. The DMA single outputs from the `DW_apb_i2c` are not sampled by the `DW_ahb_dmac` otherwise. This is illustrated in the following example. Consider first an example where the receive FIFO channel of the `DW_apb_i2c` is as follows: `DMA.CTLx.SRC_MSIZE = I2C.IC_DMA_RDLR + 1 = 4`  $\text{DMA.CTLx.BLOCK_TS} = 12$  For the example in Figure 15.37 , with the block size set to 12, the `dma_rx_req` signal is asserted when four data items are present in the receive FIFO. The `dma_rx_req` signal is asserted three times during the `DW_apb_i2c` serial transfer, ensuring that all 12 data items are read by the `DW_ahb_dmac`. All DMA requests read a block of data items and no single DMA transactions are required. This block transfer is made up of three burst transactions. Now, for the following block transfer: `DMA.CTLx.SRC_MSIZE = I2C.IC_DMA_RDLR + 1 = 4`  $\text{DMA.CTLx.BLOCK_TS} = 15$  The first 12 data items are transferred as already described using three burst transactions. But when the last three data frames enter the receive FIFO, the `dma_rx_req` signal is not activated because the FIFO level is below the watermark level. The `DW_ahb_dmac` samples `dma_rx_single` and completes the DMA block

transfer using three single transactions. The block transfer is made up of three burst transactions followed by three single transactions. Figure 15.44 shows a single transaction. The handshaking loop is as follows: `dma_tx_single/dma_rx_single` asserted by `DW_apb_i2c` -> `dma_tx_ack/dma_rx_ack` asserted by `DW_ahb_dmac` -> `dma_tx_single/dma_rx_single` de-asserted by `DW_apb_i2c` -> `dma_tx_ack/dma_rx_ack` de-asserted by `DW_ahb_dmac`.



**Figure 15.44: Single Transaction**

Figure 15.45 shows a burst transaction, followed by three back-to-back single transactions, where the `hclk` frequency is twice the `pclk` frequency.



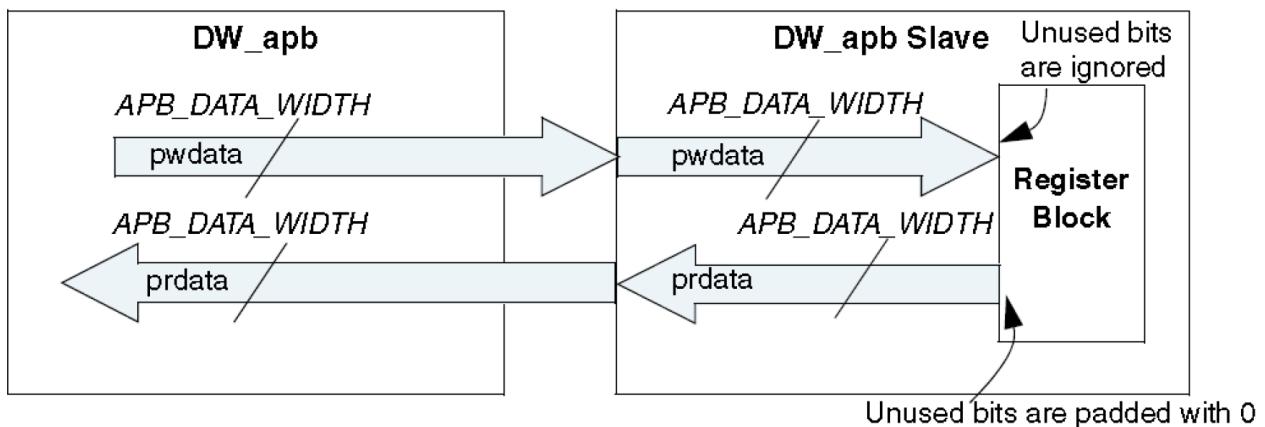
**Figure 15.45:** Burst Transaction + 3 Back-to-Back Singles -  $hclk = 2 * pclk$

#### Note

The single transaction request signals, `dma_tx_single` and `dma_rx_single`, are generated in the `DW_apb_i2c` on the `pclk` edge and sampled in `DW_ahb_dmac` on `hclk`. The acknowledge signals, `dma_tx_ack` and `dma_rx_ack`, are generated in the `DW_ahb_dmac` on the `hclk` edge `hclk` and sampled in the `DW_apb_i2c` on `pclk`. The handshaking mechanism between the `DW_ahb_dmac` and the `DW_apb_i2c` supports quasi-synchronous clocks; that is, `hclk` and `pclk` must be phase aligned and the `hclk` frequency must be a multiple of `pclk` frequency.

### 15.2.18 APB Interface

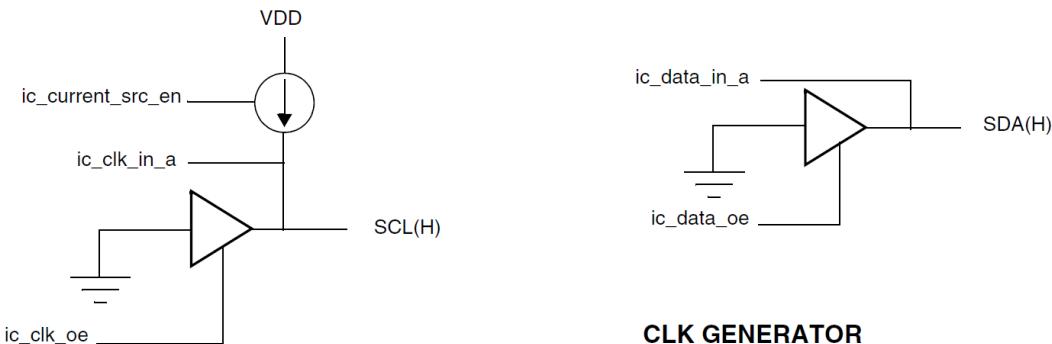
The host processor accesses data, control, and status information on the `DW_apb_i2c` peripheral through the AMBA APB 2.0 interface. This peripheral supports APB data bus widths of 8, 16, or 32 bits, which is set with the `APB_DATA_WIDTH` parameter. Figure 15.46 shows the read/write buses between the `DW_apb` and the APB slave.



**Figure 15.46:** Read/Write Buses Between the `DW_apb` and an APB Slave

### 15.2.19 I/O Connections

As illustrated in Figure 15.47, the I2C interface consists of two wires, a clock (SCL) and data (SDA). For high-speed systems, the names are SCLH and SDAH. For high-speed mode, a current source pull-up may be used on the SCLH line. It is enabled during some active master transactions. The SDA and SDAH connections are the same at any speed. There are no special connections required for the DesignWare APB slave interface side of the DW\_apb\_i2c.



**Figure 15.47:** I/O Connection to I2C Interface

### 15.2.20 DW\_apb\_i2c Registers

The "Register Descriptions" list all the registers available in DW\_apb\_i2c. Note that there are references to both hardware parameters and software registers throughout the chapter "Register Descriptions". Parameters and many of the register bits are prefixed with an IC\_\*. However, the software register bits are distinguished in "Register Descriptions" by italics. For instance, IC\_MAX\_SPEED\_MODE is a hardware parameter and configured once using Synopsys coreConsultant, whereas the IC\_SLAVE\_DISABLE bit in the IC\_CON register controls whether I2C has its slave disabled. An address definition (memory map) C header file is shipped with the DW\_apb\_i2c component. You can use this header file when the DW\_apb\_i2c is programmed in a C environment.

#### Note

A read operation to an address location that contains unused bits results in a 0 value being returned on each of the unused bits.

#### 15.2.20.1 Registers and Field Descriptions

Registers in DW\_apb\_i2c are on the pclk domain, but status bits reflect actions that occur in the ic\_clk domain. Therefore, there is delay when the pclk register reflects the activity that occurred on the ic\_clk side. Some registers may be written only when the DW\_apb\_i2c is disabled, programmed by the IC\_ENABLE register. Software should not disable the DW\_apb\_i2c while it is active. If the DW\_apb\_i2c is in the process of transmitting when it is disabled, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. The slave continues receiving until the remote master aborts the transfer, in which case the DW\_apb\_i2c could be disabled. Registers that cannot be written to when the DW\_apb\_i2c is enabled are indicated in their descriptions. Unless the clocks pclk and ic\_clk are identical (IC\_CLK\_TYPE = 0), there is a two-register delay for synchronous and asynchronous modes.

### 15.2.20.2 Operation of Interrupt Registers

Table 15.10 lists the operation of the DW\_apb\_i2c interrupt registers and how they are set and cleared. Some bits are set by hardware and cleared by software, whereas other bits are set and cleared by hardware.

Interrupt Bit Fields	Set by Hardware/ Cleared by Software	Set and Cleared by Hardware
MST_ON_HOLD	x	v
RESTART_DET	v	x
GEN_CALL	v	x
START_DET	v	x
STOP_DET	v	x
ACTIVITY	v	x
RX_DONE	v	x
TX_ABRT	v	x
RD_REQ	v	x
TX_EMPTY	x	v
TX_OVER	v	x
RX_FULL	x	v
RX_OVER	v	x
RX_UNDER	v	x

Table 15.10: Clearing and Setting of Interrupt Registers

Figure 15.48 shows the operation of the interrupt registers where the bits are set by hardware and cleared by software.

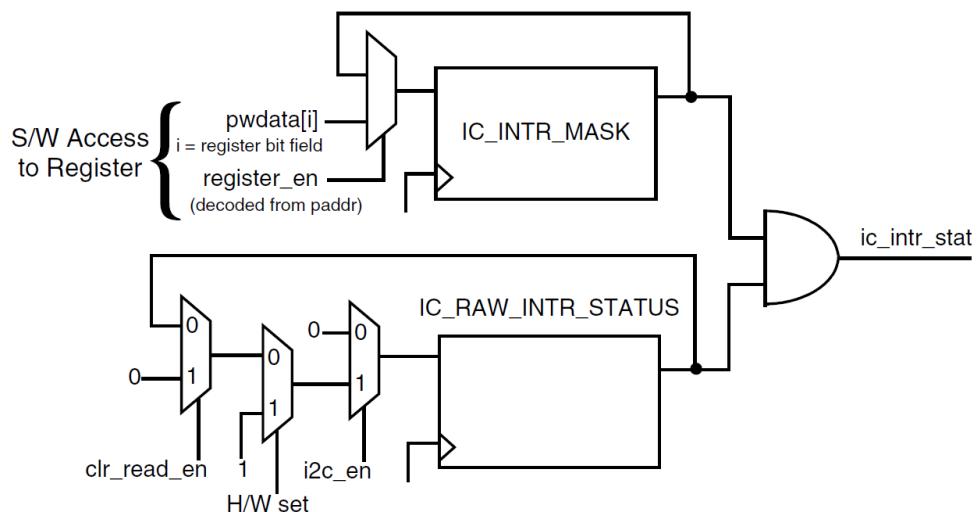


Figure 15.48: Interrupt Scheme

## 15.3 Parameter Descriptions

This chapter details all the configuration parameters. You can use the coreConsultant GUI configuration reports to determine the complete configuration state of the core. Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports. These tables define all of the user configuration options for this component.

- Top Level Parameters - I2C Version 3.0 Features - SMBus Features - I2C Version 6.0 Features

### 15.3.1 Top Level Parameters

Label	Description
System Configuration	
APB data bus width	Width of the APB data bus. 32
Device Configuration	
Highest speed I2C mode supported	<p>Maximum I2C mode supported. Controls the reset value of the SPEED bit field [2:1] of the I2C Control Register (IC_CON). Count registers are used to generate the outgoing clock SCL on the I2C interface. For speed modes faster than the configured maximum speed mode, the corresponding registers are not present in the top-level RTL.</p> <p>For unsupported speed modes those registers are not present as described below.</p> <ul style="list-style-type: none"> <li>- If this parameter is set to "Standard Mode" then the IC_FS_SCL_*, IC_HS_MADDR, and IC_HS_SCL_* registers are not present.</li> <li>- If this parameter is set to "Fast Mode" then the IC_HS_MADDR, and IC_HS_SCL_* registers are not present.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- Fast Mode or Fast Mode Plus (0x2)</li> </ul>
Has I2C default slave address of?	Reset Value of DW_apb_i2c Slave Address. Controls the reset value of Register (IC_SAR). Value: 0x055
Has I2C default target slave address of?	Reset value of DW_apb_i2c target slave address. Controls the reset value of the IC_TAR bit field (9:0) of the I2C Target Address Register (IC_TAR). The default values cannot be any of the reserved address locations: 0x00 to 0x07 or 0x78 to 0x7f. Value: 0x055
Is an I2C Master?	<p>Controls whether DW_apb_i2c has its master enabled to be a master after reset. This parameter controls the reset value of bit 0 of the I2C Control Register (IC_CON). To enable the component to be a master, you must write a 1 in bit 0 of the IC_CON register.</p> <p>Note: If this parameter is checked (1), then you must ensure that the parameter IC_SLAVE_DISABLE is checked (1) as well.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- true (0x1)</li> </ul>

Table 15.11: Top Level Parameters (Continued on next page)

continued from previous page	
Disable Slave after reset?	<p>Controls whether DW_apb_i2c has its slave enabled or disabled after reset. If checked, the DW_apb_i2c slave interface is disabled after reset. The slave also can be disabled by programming a 1 into IC_CON[6]. By default the slave is enabled.</p> <p>Note: If this parameter is unchecked (0), then you must ensure that the parameter IC_MASTER_MODE is unchecked (0) as well.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- true (0x1)</li> </ul>
Supports 10-bit addressing in slave mode?	<p>Controls whether DW_apb_i2c slave supports 7 or 10 bit addressing on the I2C interface after reset when acting as a slave. Controls reset value of part of Register IC_CON. The DW_apb_i2c module will respond to this number of address bits when acting as a slave; it can be reprogrammed by software.</p> <p>Note: The 10-bit Addressing mode is not supported in SMBus Mode.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- true (0x1)</li> </ul>
Supports 10-bit addressing in master mode?	<p>Controls whether DW_apb_i2c supports 7 or 10 bit addressing on the I2C interface after reset when acting as a master. Controls reset value of part of Register IC_CON. Master generated transfers will use this number of address bits.</p> <p>Additionally, it can be reprogrammed by software by writing to the IC_CON register.</p> <p>Note: The 10-bit Addressing mode is not supported in SMBus Mode.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- true (0x1)</li> </ul>
Depth of transmit buffer is?	<p>Depth of transmit buffer. The buffer is 9 bits wide; 8 bits for the data, and 1 bit for the read or write command.</p> <p>Value: 8</p>
Depth of receive buffer is?	<p>Depth of receive buffer, the buffer is 8 bits wide.</p> <p>Value: 8</p>
Transmit buffer threshold value is?	<p>Reset value for threshold level of transmit buffer. This parameter controls the reset value of the I2C Transmit FIFO Threshold Level Register (IC_TX_TL).</p> <p>Value: 0x0</p>
Receive buffer threshold value is?	<p>Reset value for threshold level of receive buffer. This parameter controls the reset value of the I2C Receive FIFO Threshold Level Register (IC_RX_TL).</p> <p>Value: 0x0</p>
Allow re-start conditions to be sent when acting as a master?	<p>Controls the reset value of bit 5 (IC_RESTART_EN) in the IC_CON register. By default, this parameter is checked, which allows RESTART conditions to be sent when DW_apb_i2c is acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several I2C operations. When the RESTART is disabled, the DW_apb_i2c master is incapable of performing the following functions:</p> <ul style="list-style-type: none"> <li>- Sending a START BYTE</li> <li>- Performing any high-speed mode operation</li> <li>- Performing direction changes in combined format mode</li> <li>- Performing a read operation with a 10-bit address</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- true (0x1)</li> </ul>

**Table 15.11:** Top Level Parameters (Continued on next page)

		continued from previous page
Hardware reset value for IC_SDA_SETUP register	Determines the reset value for the register IC_SDA_SETUP, which in turn controls the time delay - in terms of number of ic_clk clock periods - introduced in the rising edge of SCL, relative to SDA changing when a read-request is serviced. The relevant I2C requirement is t[su:DAT] as detailed in the I2C Bus Specifications. Value: 0x64	
Hardware reset value for IC_SDA_HOLD register	Determines the reset value for the register IC_SDA_HOLD, which in turn controls the SDA hold time implemented by DW_apb_i2c (when transmitting or receiving, as either master or slave) as a master/slave transmitter or Master/Slave Reciever). The relevant I2C requirement is t[HD:DAT] as detailed in the I2C Bus Specifications. The programmed SDA hold time as transmitter cannot exceed at any time the duration of the low part of scl. Therefore it is recommended that the configured default value should not be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the scl period measured in ic_clk cycles, for the maximum speed mode the component is configured for. Values: 0x000001	
IC_ACK_GENERAL_CALL set to acknowledge I2C general calls on reset	This parameter determines the reset value for the register IC_ACK_GENERAL_CALL, which in turn controls whether I2C general call addresses are responded or not. Values: - true (0x1)	
External Configuration		
Include DMA handshaking interface signals?	Configures the inclusion of DMA handshaking interface signals. When checked, includes the DMA handshaking interface signals at the top-level I/O. For more information about these signals, see "Signal Descriptions" in data book. Values: - false (0x0)	
Single Interrupt output port present?	If unchecked, each interrupt source has its own output. If checked, all interrupt sources are combined into a single output. Values: - true (0x1)	
Polarity of Interrupts is active high?	Configures the active level of the output interrupt lines. Values: - true (0x1)	
Internal Configuration		
<b>Table 15.11:</b> Top Level Parameters (Continued on next page)		

		continued from previous page
Add Encoded Parameters	<p>Adding the encoded parameters gives firmware an easy and quick way of identifying the DesignWare component within an I/O memory map. Some critical design-time options determine how a driver should interact with the peripheral. There is a minimal area overhead by including these parameters. Allows a single driver to be developed for each component which will be self-configurable.</p> <p>When bit 7 of the IC_COMP_PARAM_1 is read and contains a '1' the encoded parameters can be read via software. If this bit is a '0' then the entire register is '0' regardless of the setting of any of the other parameters that are encoded in the register's bits. For details about this register, see the IC_COMP_PARAM_1 register.</p> <p>Note: Unique drivers must be developed for each configuration of the DW_apb_i2c. Based on the configuration, the registers in the IP can differ; thus the same driver cannot be used with different configurations of the IP.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- true (0x1)</li> </ul>	
Specify clock counts directly instead of supplying clock frequency?	<p>Determines whether *CNT values are provided directly or by specifying the ic_clk clock frequency and letting coreConsultant (or coreAssembler) calculate the count values.</p> <p>When this parameter is checked, the reset values of the *CNT registers are specified by the corresponding *COUNT configuration parameters which may be user-defined or derived (see standard, fast, fast mode plus, and high speed mode parameters later in this table).</p> <p>When unchecked (default setting), the reset values of the *CNT registers are calculated from the configuration parameter IC_CLOCK_PERIOD.</p> <p>Note: For fast mode plus, reprogram the IC_FS_SCL_*CNT register to achieve the required data rate when unchecked.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>	
Hard code the count values for each mode?	<p>By checking this parameter, the *CNT registers are set to read only. Unchecking this parameter (default setting) allows the *CNT registers to be writable. Regardless of the setting, the *CNT registers are always readable and have reset values from the corresponding *COUNT configuration parameters, which may be user defined or derived (see standard, fast, fast mode plus, or high speed mode parameters later in this table).</p> <p>Note: Since the DW_apb_i2c uses the same high and low count registers for fast mode and fast mode plus operation, if this parameter is checked (1) the IC_FS_SCL_*CNT registers are hard coded to either one of the fast mode and fast mode plus. Consequently, DW_apb_i2c can operate in either fast mode or fast mode plus, but not in both modes simultaneously.</p> <p>For fast mode plus, it is recommended that this parameter be Unchecked (0).</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>	

**Table 15.11:** Top Level Parameters (Continued on next page)

continued from previous page	
ic_clk has a period of? (ns integers only)	Specifies the period of incoming ic_clk, used to generate outgoing I2C interface SCL clock. (ns integers only) When the count values are used to generate the IC_CLOCK_PERIOD then the IC_MAX_SPEED_MODE setting determines the actual period IC_MAX_SPEED_MODE = Standard => 500ns IC_MAX_SPEED_MODE = Fast => 100ns IC_MAX_SPEED_MODE = High => 10ns IC_ULTRA_FAST_MODE = 1 => 25ns Note: For fast mode plus, user has to reprogram the IC_FS_SCL_*CNT register to achieve required data rate. Values: 100
Relationship between pclk and ic_clk is?	Specifies the relationship between pclk and ic_clk Identical (0): clocks are identical; no meta-stability flops used for data passing between clock domains. Asynchronous (1): clocks may be completely asynchronous to each other, meta- stability flops are required for data passing between clock domains. Values: - Identical (0x0)
Enable Async FIFO Mode?	This parameter controls whether DW_apb_i2c consist of Asynchronous or Synchronous FIFO's for the Transmit and Receive Data Buffers. Values: - false (0x0)
Spike Suppression Configuration	
Maximum length (in ic_clk cycles) of suppressed spikes in Standard Mode, Fast Mode, and Fast Mode Plus	Reset value of maximum suppressed spike length register in Standard Mode, Fast Mode, and Fast Mode Plus modes (IC_FS_SPKLEN Register). Spike length is expressed in ic_clk cycles and this value is calculated based on the value of IC_CLOCK_PERIOD. Values: 0x5
Additional Features	
Allow dynamic updating of the TAR address?	When checked, allows the IC_TAR register to be updated dynamically. Setting this parameter affects the operation of DW_apb_i2c when it is in master mode. For more details, see "Master Mode Operation". Values: - false (0)
Enable register to generate NACKs for data received by Slave?	Enables an additional register which controls whether the DW_apb_i2c generates a NACK after a data byte has been transferred to it. This NACK generation only occurs when the DW_apb_i2c is a Slave-Receiver. If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence, the data transfer is aborted. Also, the data received is not pushed to the receive buffer. When the register is set to a value of 0, it generates NACK/ACK depending on normal criteria. If this option is selected, the default value of the register IC_SLV_DATA_NACK_ONLY is always 0. The register must be explicitly programmed to a value of 1 if NACKs are to be generated. The register can only be written to successfully if DW_apb_i2c is disabled (IC_ENABLE[0] = 0) or the slave part is inactive (IC_STATUS[6] = 0). Values: - false (0x0)

Table 15.11: Top Level Parameters (Continued on next page)

continued from previous page	
Hold transfer when Tx FIFO is empty	<p>If this parameter is set, the master will only complete a transfer - that is issues a STOP - when it finds a Tx FIFO entry tagged with a Stop bit. If the Tx FIFO becomes empty and the last byte does not have the Stop bit set, the master stalls the transfer by holding the SCL line low.</p> <p>If this parameter is not set, the master completes a transfer when the Tx FIFO is empty. In SMBus Mode (IC_SMBUS=1), IC_EMPTYFIFO_HOLD_MASTER_EN should be always enabled.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0)</li> </ul>
When Receive Fifo is physically full, Hold the bus till Receive fifo has space available?	<p>When the Rx FIFO is physically full to its RX_BUFFER_DEPTH, this parameter provides a hardware method to hold the bus till Rx FIFO data is read out and there is a space available in the FIFO. This parameter can be used when DW_apb_i2c is either a slave-receiver (that is, data is written to the device) or a master-receiver (that is, the device reads data from a slave).</p> <p>Note: If this parameter is checked, then the RX_OVER interrupt is never set to 1 as the criteria to set this interrupt is never met. The RX_OVER interrupt can be found in IC_INTR_STAT and IC_RAW_INTR_STAT registers. It is also an optional output signal, ic_rx_over_intr(_n).</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Enable restart detect interrupt in slave mode?	<p>When checked, allows the slave to detect and issue the restart interrupt when slave is addressed. Setting this parameter affects the operation of DW_apb_i2c only when it is in slave mode. This controls the "RESTART_DET" bit in the IC_RAW_INTR_STAT, IC_INTR_MASK, IC_INTR_STAT, and IC_CLR_RESTART_DET registers. This also controls the ic_restart_det_intr(_n) and ic_intr(_n) signals.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Generate STOP_DET interrupt only if Master is active?	<p>Controls whether DW_apb_i2c generates STOP_DET interrupt when master is active:</p> <ul style="list-style-type: none"> <li>- Checked (1): Allows the master to detect and issue the stop interrupt when master is active.</li> <li>- Unchecked (0): The master always detects and issues the stop interrupt irrespective of whether it is active.</li> </ul> <p>This parameter affects the operation of DW_apb_i2c when it is in master mode. This controls the STOP_DET bit of the IC_RAW_INTR_STAT, IC_INTR_MASK, IC_INTR_STAT and IC_CLR_STOP_DET registers. This also controls the ic_stop_det_intr(_n) and ic_intr(_n) signals.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Include Status bits to indicate the reason for clock stretching?	<p>If this parameter is set, the DW_apb_i2c consists of status bits indicating the reason for clock stretching in the IC_STATUS Register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Include programmable bit for blocking Master commands?	<p>Controls whether DW_apb_i2c transmits data on I2C bus as soon as data is available in Tx FIFO. When checked, allows the master to hold the transmission of data on I2C bus when Tx FIFO has data to transmit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>

**Table 15.11: Top Level Parameters (Continued on next page)**

continued from previous page	
Include First data byte indication in IC_DATA_CMD register?	<p>Controls whether DW_apb_i2c generates FIRST_DATA_BYTE status bit in IC_DATA_CMD register. When checked, the master/slave receiver to set the FIRST_DATA_BYTE status bit in IC_DATA_CMD register to indicate whether the data present in IC_DATA_CMD register is first data byte after the address phase of a receive transfer.</p> <p>Note: In the case when APB_DATA_WIDTH is set to 8, you must perform two APB reads to the IC_DATA_CMD register to get status on bit 11.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Avoid Rx FIFO Flush on Tranmsit Abort?	<p>This Parameter controls the Rx FIFO Flush during the Transmit Abort. If this parameter is checked(1), only the Tx FIFO is flushed (not the Rx FIFO) Flush on the Transmit Abort. If this parameter is unchecked(0), both Tx FIFO and Rx FIFO are flushed on Transmit Abort.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Enable IC_CLK Frequency Reduction?	<p>This parameter is used to reduce the system clock frequency (ic_clk) by reducing the internal latency required to generate the high period and low period of the SCL line.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>

**Table 15.11:** Top Level Parameters**Table 15.12:** I2C Version 3.0 Features Parameters (Continued on next page)

continued from previous page

### 15.3.2 I2C Version 3.0 Features Parameters

Label	Description
I2C 3.0 Features	
Include Bus Clear feature?	<p>This parameter will enable the Bus clear feature for the DW_apb_i2c core. If this parameter is set:</p> <ul style="list-style-type: none"> <li>- If an SDA line is stuck at low for IC_SDA_STUCK_LOW_TIMEOUT period of ic_clk, DW_apb_i2c master generates a master transmit abort (IC_TX_ABRT_SOURCE[17]: ABRT_SDA_STUCK_AT_LOW) to indicate SDA stuck at low.</li> </ul> <p>User can enable the SDA_STUCK_RECOVERY_EN (IC_ENABLE[3]) register bit to recover the SDA by sending at most 9 SCL clocks.</p> <p>If SDA line is recovered, then the master generates a STOP and auto clear the 'SDA_STUCK_RECOVERY_EN' register bit and resume the normal I2C transfers.</p> <p>If an SDA line is not recovered, then the master auto clears the SDA_STUCK_RECOVERY_EN register bit and asserts the SDA_STUCK_NOT_RECOVERED (IC_STATUS[12]) status bit to indicate the SDA is not recovered after sending 9 SCL clocks which intimate the user for system reset.</p> <ul style="list-style-type: none"> <li>- If SCL line is stuck at low for IC_SCL_STUCK_LOW_TIMEOUT period of ic_clk, DW_apb_i2c Master will generate an SCL_STUCK_AT_LOW (IC_INTR_RAW_STATUS[14]) interrupt to intimate the user for system reset.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>
Enable DEVICE-ID feature?	<p>If this Parameter is enabled, the DW_apb_i2c slave includes a 24-bit IC_DEVICE_ID Register to store the value of Device-ID and transmits whenever master is requested.</p> <p>The Master mode includes a DEVICE_ID bit 13 in IC_TAR register to initiate the Device ID read for a particular slave address mentioned in IC_TAR[6:0] register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>

Table 15.12: I2C Version 3.0 Features Parameters

### 15.3.3 SMBus Features Parameters

Label	Description
I2C System Management Bus Features	
Enable SMBus Mode?	<p>Controls whether DW_apb_i2c Master/Slave supports SMBus mode. If checked, the DW_apb_i2c includes the SMBus mode related registers, real-time checks, timeout interrupts, and SMBus optional signals.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>- If this parameter is selected (1), then the user can set the parameter IC_MAX_SPEED_MODE to Standard mode(1) or Fast Mode/Fast Mode Plus (2).</li> <li>- The 10-bit Addressing mode is not supported in SMBus Mode.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- false (0x0)</li> </ul>

Table 15.13: SMBus Features Parameters (Continued on next page)

continued from previous page

**Table 15.13:** SMBus Features Parameters

#### 15.3.4 I2C Version 6.0 Features Parameters

Label	Description
I2C System Management Bus Features	
Enable SMBus Mode?	<p>Controls whether DW_apb_i2c Master/Slave supports SMBus mode. If checked, the DW_apb_i2c includes the SMBus mode related registers, real-time checks, timeout interrupts, and SMBus optional signals.</p> <p>Note:</p> <ul style="list-style-type: none"><li>- If this parameter is selected (1), then the user can set the parameter IC_MAX_SPEED_MODE to Standard mode(1) or Fast Mode/Fast Mode Plus (2).</li><li>- The 10-bit Addressing mode is not supported in SMBus Mode.</li></ul> <p>Values:</p> <ul style="list-style-type: none"><li>- false (0x0)</li></ul>

**Table 15.14:** I2C Version 6.0 Features Parameters

## 15.4 Signal Descriptions

This chapter details all possible I/O signals in the core. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the core. It is for reference purposes only.

When you configure the core in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

In addition to describing the function of each signal, the signal descriptions in this chapter include the following information:

Active State: Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the in-active state (opposite of the active state).

Registered: Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of No does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the controller. A value of N/A indicates that this information is not provided for this IP title.

Synchronous to: Indicates which clock(s) in the IP sample this input (drive for an output) when considering all possible configurations. A particular configuration might not have all of the clocks listed. This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.

Exists: Name of configuration parameter(s) that populates this signal in your configuration.

Validated by: Assertion or de-assertion of signal(s) that validates the signal being described.

The I/O signals are grouped as follows:

- Interrupts
- I2C Interface (Master/Slave)
- APB Slave Interface
- DMA Interface
- SMBus Interface
- I2C Debug

### 15.4.1 Interrupts Signals

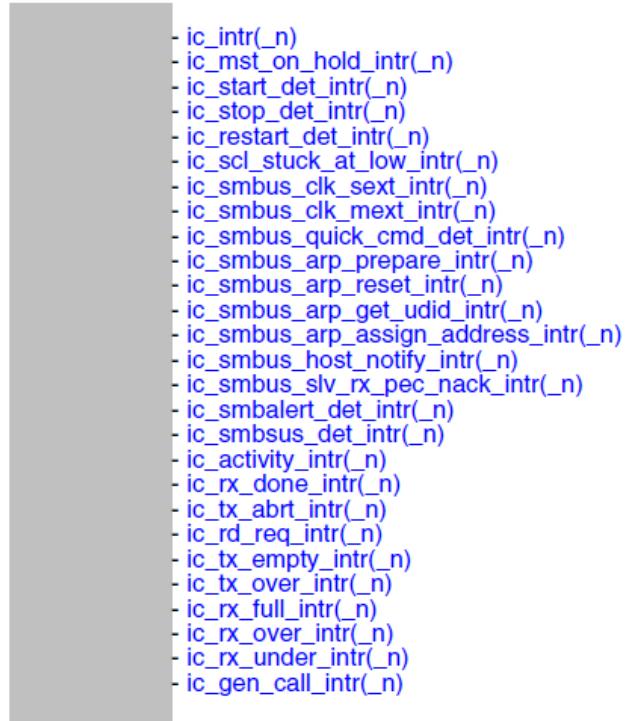


Figure 15.49

Port Name	I/O	Description
ic_mst_on_hold_intr(_n)	O	<p>Optional. Master on hold I2C interrupt. This signal is included on the interface when the configuration parameters I2C_DYNAMIC_TAR_UPDATE and IC_EMPTYFIFO_HOLD_MASTER_EN are checked (1) and the configuration parameter IC_INTR_IO is unchecked (0), indicating that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; I2C_DYNAMIC_TAR_UPDATE==1 &amp; IC_EMPTYFIFO_HOLD_MASTER_EN==1</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High when IC_INTR_POL=1 otherwise Low</p> <p>Synchronous to: pcclk</p> <p>Registered: Yes</p>

Table 15.15: Interrupts Signals (Continued on next page)

continued from previous page		
ic_start_det_intr(_n)	O	<p>Optional. Start condition detect on I2C interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_stop_det_intr(_n)	O	<p>Optional. Stop condition detect on I2C interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_restart_det_intr(_n)	O	<p>Optional. Restart condition detect on I2C interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SLV_RESTART_DET_EN==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_scl_stuck_at_low_intr(_n)	O	<p>Optional. SCL Stuck condition detect on I2C interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_BUS_CLEAR_FEATURE==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_smbus_clk_sext_intr(_n)	O	<p>Optional. SMBUS Slave clock extend timeout detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>

**Table 15.15:** Interrupts Signals (Continued on next page)

continued from previous page		
ic_smbus_clk_mext_intr(_n)	O	<p>Optional. SMBUS Master clock extend timeout detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_smbus_quick_cmd_det_intr(_n)	O	<p>Optional. SMBUS ARP Quick Command detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_smbus_arp_prepare_intr(_n)	O	<p>Optional. SMBUS ARP Prepare Command detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_ARP==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_smbus_arp_reset_intr(_n)	O	<p>Optional. SMBUS ARP Reset Command detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_ARP==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_smbus_arp_get_udid_intr(_n)	O	<p>Optional. SMBUS ARP Get UDID Command detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_ARP==1  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>

**Table 15.15:** Interrupts Signals (Continued on next page)

continued from previous page		
ic_smbus_arp_assign_address_intr(_n)	O	<p>Optional. SMBUS ARP Assign Command detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_ARP==1 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pcik Registered: Yes</p>
ic_smbus_host_notify_intr(_n)	O	<p>Optional. SMBUS ARP Host Notify Command detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS==1 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pcik Registered: Yes</p>
ic_smbus_slv_rx_pec_nack_intr(_n)	O	<p>Optional. SMBUS ARP Slave Received incorrect PEC Byte and generated Nack interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_ARP==1 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pcik Registered: Yes</p>
ic_smbalert_det_intr(_n)	O	<p>Optional. SMBUS Alert detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_SUSPEND_ALERT==1 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pcik Registered: Yes</p>
ic_smbsus_det_intr(_n)	O	<p>Optional. SMBUS Suspend detect interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_SMBUS_SUSPEND_ALERT==1 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pcik Registered: Yes</p>

**Table 15.15:** Interrupts Signals (Continued on next page)

continued from previous page		
ic_activity_intr(_n)	O	<p>Optional. I2C activity interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_rx_done_intr(_n)	O	<p>Optional. Receive done interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_ULTRA_FAST_MODE==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_tx_abrt_intr(_n)	O	<p>Optional. Transmit abort interrupt.</p> <p>Exists: IC_INTR_IO==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_rd_req_intr(_n)	O	<p>Optional. Slave read request interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>Exists: IC_INTR_IO==0 &amp; IC_ULTRA_FAST_MODE==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>
ic_tx_empty_intr(_n)	O	<p>Optional. Transmit buffer empty interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>When bit 0 of the IC_ENABLE register is 0, the TX FIFO is flushed and held in reset, where it looks like it has no data within it. The ic_tx_empty_n bit is raised when bit 0 of the IC_ENABLE register is 0, provided there is activity in the master or slave state machines. When there is no longer activity, then this interrupt bit is masked with ic_en.</p> <p>Exists: IC_INTR_IO==0  Power Domain: SINGLE_DOMAIN  Active State: High when IC_INTR_POL=1 otherwise Low  Synchronous to: pcik  Registered: Yes</p>

**Table 15.15:** Interrupts Signals (Continued on next page)

continued from previous page		
ic_tx_over_intr(_n)	O	<p>Optional. Transmit buffer overflow interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>When the module is disabled, this interrupt keeps its level until the master or slave state machines go into idle and bit 0 of the IC_ENABLE register is 0. When ic_en goes to 0, this interrupt is cleared.</p> <p>Exists: IC_INTR_IO==0 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pclk Registered: Yes</p>
ic_rx_full_intr(_n)	O	<p>Optional. Receive buffer full interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>When bit 0 of the IC_ENABLE register is 0, the RX FIFO is flushed and held in resetthe RX FIFO is not fullso this ic_rx_full_intr_n bit is cleared once the ic_enable bit is programmed with a 0, regardless of the activity that continues.</p> <p>Exists: IC_INTR_IO==0 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pclk Registered: Yes</p>
ic_rx_over_intr(_n)	O	<p>Optional. Receive buffer overflow interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>When the module is disabled, this interrupt keeps its level until the master or slave state machines go into idle and bit 0 of the IC_ENABLE register is 0. When ic_en goes to 0, this interrupt is cleared.</p> <p>Exists: IC_INTR_IO==0 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pclk Registered: Yes</p>
ic_rx_under_intr(_n)	O	<p>Optional. Receive buffer underflow interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O.</p> <p>When the module is disabled, this interrupt keeps its level until the master or slave state machines go into idle and bit 0 of the IC_ENABLE register is 0. When ic_en goes to 0, this interrupt is cleared.</p> <p>Exists: IC_INTR_IO==0 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pclk Registered: Yes</p>

Table 15.15: Interrupts Signals (Continued on next page)

continued from previous page		
ic_gen_call_intr(_n)	O	Optional. General Call received interrupt. This signal is included on the interface when the configuration IC_INTR_IO parameter is unchecked (0), which indicates that individual interrupt lines appear on the I/O. Exists: IC_INTR_IO==0 Power Domain: SINGLE_DOMAIN Active State: High when IC_INTR_POL=1 otherwise Low Synchronous to: pcik Registered: Yes

**Table 15.15:** Interrupts Signals**Table 15.16:** I2C Interface (Master/Slave) Signals (Continued on next page)

	continued from previous page
--	------------------------------

### 15.4.2 I2C Interface (Master/Slave) Signals



**Figure 15.50**

Port Name	I/O	Description
ic_current_src_en	O	<p>Optional. Current source pull-up. Controls the polarity of the current source pull-up on the SCLH. This pull-up is used to shorten the rise time on SCLH by activating an user-supplied external current source pull-up circuit. It is disabled after a RESTART condition and after each A/A bit when acting as the active master.</p> <p>This signal enables other devices to delay the serial transfer by stretching the LOW period of the SCLH signal. The active master re-enables its current source pull-up circuit again when all devices have released and the SCLH signal reaches high level, therefore, shortening the last part of the SCLH signal's rise time.</p> <p>Exists: (IC_MAX_SPEED_MODE==3) Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes</p>
ic_clk	I	<p>Peripheral clock. DW_apb_i2c runs on this clock and is used to clock transfers in standard, fast, and high-speed mode.</p> <p>Note: ic_clk frequency must be greater than or equal to pclk frequency.</p> <p>Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: The configuration parameter IC_CLK_TYPE indicates the relationship between pclk and ic_clk. It can be asynchronous (1) or identical (0). Registered: N/A</p>

**Table 15.16:** I2C Interface (Master/Slave) Signals (Continued on next page)

continued from previous page		
ic_clk_in_a	I	<p>In (IC_ULTRA_FAST_MODE = 0) mode - Incoming I2C clock. This is the input SCL signal. Double-registered for metastability synchronization.</p> <p>Note: DW_apb_i2c provides filtering on the SDA (ic_data_in_a) and SCL (ic_clk_in_a) inputs, suppressing noise and signal spikes with durations less than one ic_clk period.</p> <p>In Ultra-Fast(IC_ULTRA_FAST_MODE = 1) mode - Incoming I2C clock. This is the input SCL signal. Double-registered for metastability synchronization.</p> <p>Note: DW_apb_i2c provides filtering on the SDA (ic_data_in_a) and SCL (ic_clk_in_a) inputs, suppressing noise and signal spikes with durations less than one ic_clk period. This signal is used as USCL input for slave device.</p> <p>Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: This signal is asynchronous to ic_clk. Registered: Yes</p>
ic_data_in_a	I	<p>In (IC_ULTRA_FAST_MODE = 0) mode - Incoming I2C Data. It is the input SDA signal. Double-registered for metastability synchronization.</p> <p>Note: DW_apb_i2c provides filtering on the SDA (ic_data_in_a) and SCL (ic_clk_in_a) inputs, suppressing noise and signal spikes with durations less than one ic_clk period.</p> <p>In Ultra-Fast(IC_ULTRA_FAST_MODE = 1) mode - Incoming I2C Data. It is the input SDA signal. Double-registered for metastability synchronization.</p> <p>Note: DW_apb_i2c provides filtering on the SDA (ic_data_in_a) and SCL (ic_clk_in_a) inputs, suppressing noise and signal spikes with durations less than one ic_clk period. This signal is used as USDA input for slave device.</p> <p>Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: This signal is asynchronous to ic_clk. Registered: Yes</p>
ic_rst_n	I	<p>I2C reset. Used to reset flip-flops that are clocked by the ic_clk clock.</p> <p>Note: This signal does not reset DW_apb_i2c control, configuration, and status registers.</p> <p>Exists: Always Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: The signal is asserted asynchronously, but is deasserted synchronously after the rising edge of ic_clk. The synchronization must be provided external to this component. Registered: N/A</p>

Table 15.16: I2C Interface (Master/Slave) Signals (Continued on next page)

continued from previous page		
------------------------------	--	--

ic_clk_oe	O	In (IC_ULTRA_FAST_MODE = 0) mode - Outgoing I2C clock. Open drain synchronous with ic_clk. In Ultra-Fast(IC_ULTRA_FAST_MODE = 1) mode - Outgoing I2C clock, inverted. This signal is used as USCL out from master device. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
ic_data_oe	O	In (IC_ULTRA_FAST_MODE = 0) mode - Outgoing I2C Data. Open Drain Synchronous to ic_clk. In Ultra-Fast(IC_ULTRA_FAST_MODE = 1) mode - Outgoing I2C Data, inverted. This signal is used as USDA out from master device. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
ic_en	O	I2C interface enable. Indicates whether DW_apb_i2c is enabled; this signal is set to 0 when IC_ENABLE[0] is set to 0 (disabled). Because DW_apb_i2c always finishes its current transfer before turning off ic_en, this signal may be used by a clock generator to control whether the DW_apb_i2c ic_clk is active or inactive. Exists: Always Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: pclk Registered: Yes

**Table 15.16:** I2C Interface (Master/Slave) Signals**Table 15.17:** APB Slave Interface Signals (Continued on next page)

continued from previous page

### 15.4.3 APB Slave Interface Signals

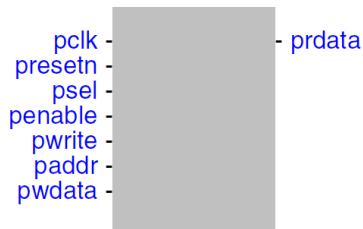


Figure 15.51

Port Name	I/O	Description
pclk	I	APB clock for the bus interface unit. Note: ic_clk frequency must be greater than or equal to pclk frequency. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: The configuration parameter IC_CLK_TYPE indicates the relationship between pclk and ic_clk. It can be asynchronous (1) or identical (0). Registered: N/A
presetn	I	An APB interface domain reset. Exists: Always Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: The signal is asserted asynchronously, but is deasserted synchronously after the rising edge of pclk. The synchronization must be provided external to this component. Registered: N/A
psel	I	APB peripheral select that lasts for two pclk cycles. When asserted, indicates that the peripheral has been selected for a read/write operation. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: pclk Registered: No
penable	I	APB enable control. Asserted for a single pclk cycle and used for timing read/write operations. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: pclk Registered: No

Table 15.17: APB Slave Interface Signals (Continued on next page)

continued from previous page		
pwrite	I	APB write control. When high, indicates a write access to the peripheral; when low, indicates a read access. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: pclk Registered: No
paddr[IC_ADDR_SLICE_LHS:0]	I	APB address bus. Uses lower 7 bits of the address bus for register decode. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: pclk Registered: No
pwdata[(APB_DATA_WIDTH-1):0]	I	APB write data bus. Driven by the bus master (DW_ahb to DW_apb bridge) during write cycles. Can be 8, 16, or 32 bits wide depending on APB_DATA_WIDTH parameter. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: pclk Registered: No
prdata[(APB_DATA_WIDTH-1):0]	O	APB readback data. Driven by the selected peripheral during read cycles. Can be 8, 16, or 32 bits wide depending on APB_DATA_WIDTH parameter. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: pclk Registered: Yes

**Table 15.17:** APB Slave Interface Signals**Table 15.18:** DMA Interface Signals (Continued on next page)

continued from previous page

#### 15.4.4 DMA Interface Signals



Figure 15.52

Port Name	I/O	Description
dma_tx_ack	I	<p>Optional. DMA Transmit Acknowledgement. Sent by the DMA Controller to acknowledge the end of each APB transfer burst to the transmit FIFO.</p> <p>Exists: (IC_HAS_DMA==1)</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High Synchronous to: pclk Registered: No</p>
dma_tx_req	O	<p>Optional. Transmit FIFO DMA Request. Asserted when the transmit FIFO requires service from the DMA Controller; that is, the transmit FIFO is at or below the watermark level.</p> <ul style="list-style-type: none"> <li>- 0 not requesting</li> <li>- 1 requesting</li> </ul> <p>Software must set up the DMA controller with the number of words to be transferred when a request is made. When using the DW_ahb_dmac, this value is programmed in the DEST_MSIZE field of the CTLx register.</p> <p>Exists: (IC_HAS_DMA==1)</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High Synchronous to: pclk Registered: Yes</p>
dma_tx_single	O	<p>Optional. DMA Transmit FIFO Single Signal. This DMA status output informs the DMA Controller that there is at least one free entry in the transmit FIFO. This output does not request a DMA transfer.</p> <ul style="list-style-type: none"> <li>- 0: Transmit FIFO is full</li> <li>- 1: Transmit FIFO is not full</li> </ul> <p>Exists: (IC_HAS_DMA==1)</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High Synchronous to: pclk Registered: Yes</p>
dma_rx_ack	I	<p>Optional. DMA Receive Acknowledgement. Sent by the DMA controller to acknowledge the end of each APB transfer burst from the receive FIFO.</p> <p>Exists: (IC_HAS_DMA==1)</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High Synchronous to: pclk Registered: No</p>

Table 15.18: DMA Interface Signals (Continued on next page)

continued from previous page		
dma_rx_req	O	<p>Optional. Receive FIFO DMA Request. Asserted when the receive FIFO requires service from the DMA Controller; that is, the receive FIFO is at or above the watermark level.</p> <ul style="list-style-type: none"> <li>- 0 not requesting</li> <li>- 1 requesting</li> </ul> <p>Software must set up the DMA controller with the number of words to be transferred when a request is made. When using the DW_ahb_dmac, this value is programmed in the SRC_MSIZE field of the CTLx register.</p> <p>Exists: (IC_HAS_DMA==1)  Power Domain: SINGLE_DOMAIN  Active State: High Synchronous to: pclk Registered: Yes</p>
dma_rx_single	O	<p>Optional. DMA Receive FIFO Single Signal. This DMA status output informs the DMA Controller that there is at least one valid data entry in the receive FIFO. This output does not request a DMA transfer.</p> <ul style="list-style-type: none"> <li>- 0: Receive FIFO is empty</li> <li>- 1: Receive FIFO is not empty</li> </ul> <p>Exists: (IC_HAS_DMA==1)  Power Domain: SINGLE_DOMAIN  Active State: High Synchronous to: pclk Registered: Yes</p>

**Table 15.18:** DMA Interface Signals**Table 15.19:** SMBus Interface Signals (Continued on next page)

	continued from previous page
--	------------------------------

### 15.4.5 SMBus Interface Signals

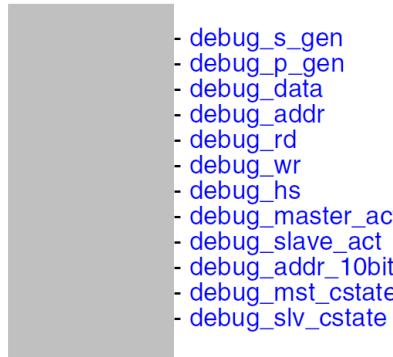


**Figure 15.53**

Port Name	I/O	Description
ic_smbsus_in_n	I	Incoming SMBus Suspend signal. This is the input SMBSUS signal. Double-registered for metastability synchronization. Exists: (IC_SMBUS_SUSPEND_ALERT==1) Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: This signal is asynchronous to pclk Registered: Yes
ic_smbalert_in_n	I	Incoming SMBus Alert signal. This is the input SMBALERT signal. Double-registered for metastability synchronization. Exists: (IC_SMBUS_SUSPEND_ALERT==1) Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: This signal is asynchronous to pclk Registered: Yes
ic_smbsus_out_n	O	Outgoing SMBus Suspend Signal. This signal is used to suspend the SMBus system, if DW_apb_i2c is used as SMBus Host. Exists: (IC_SMBUS_SUSPEND_ALERT==1) Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: pclk Registered: Yes
ic_smbalert_oe	O	Outgoing SMBus Alert Signal. This signal is used to intimate the Host that slave wants to talk, if DW_apb_i2c is used as SMBus Slave. Exists: (IC_SMBUS_SUSPEND_ALERT==1) Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: pclk Registered: Yes

**Table 15.19:** SMBus Interface Signals

### 15.4.6 I2C Debug Signals



**Figure 15.54**

Port Name	I/O	Description
debug_s_gen	O	In the master mode of operation, this signal is set to 1 when DW_apb_i2c is driving a START condition on the bus. Exists: Always Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: ic_clk Registered: Yes
debug_p_gen	O	In the master mode of operation, this signal is set to 1 when DW_apb_i2c is driving a STOP condition on the bus. Exists: Always Power Domain: SINGLE_DOMAIN Active State: Low Synchronous to: ic_clk Registered: Yes
debug_data	O	In the master or slave mode of operation, this signal is set to 1 when a byte of data is actively being read or written by DW_apb_i2c. This bit remains 1 until the transaction has completed. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: N/A Registered: Yes
debug_addr	O	In the master or slave mode of operation, this signal is set to 1 when the addressing phase is active on the I2C bus. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
debug_rd	O	In the master mode of operation, this signal is set to 1 whenever the master is receiving data. This bit remains 1 until the transfer is complete or until the direction changes. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes

**Table 15.20:** I2C Debug Signals (Continued on next page)

continued from previous page		
debug_wr	O	In the master mode of operation, this signal is set to 1 whenever the master is transmitting data. This bit remains 1 until the transfer is complete or the direction changes. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
debug_hs	O	In the master mode of operation, this signal is set to 1 when DW_apb_i2c is performing high-speed mode transfers. This bit is set after the high-speed master code is transmitted and remains 1 until the master leaves high-speed mode. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
debug_master_act	O	This bit is set to 1 when the master module is active. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
debug_slave_act	O	This bit is set to 1 when the slave module is active. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
debug_addr_10bit	O	In the Slave mode of operation, this signal is set if 10-bit addressing is enabled and if the slave has received a matching 10-bit address with respect to IC_SAR register. This signal is not applicable in Master Mode. Exists: Always Power Domain: SINGLE_DOMAIN Active State: High Synchronous to: ic_clk Registered: Yes
debug_mst_cstate[4:0]	O	Master FSM state vector. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: ic_clk Registered: Yes
debug_slv_cstate[3:0]	O	Slave FSM state vector. Exists: Always Power Domain: SINGLE_DOMAIN Active State: N/A Synchronous to: ic_clk Registered: Yes

Table 15.20: I2C Debug Signals

## 15.5 I2C Register Description

### 15.5.1 Register Map Summary

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)

Register	Offset	Description	Reset Value
I2C Control Register	0x00	IC_CON	0x0000_007D
I2C Target Address Register	0x04	IC_TAR	0x0000_1055
I2C Slave Address Register	0x08	IC_SAR	0x0000_0055
I2C High Speed Master Mode Code Address Register	0x0C	IC_HS_MADDR	0x0000_0001
I2C Rx/Tx Data Buffer and Command Register	0x10	IC_DATA_CMD	0x0000_0000
Standard Speed I2C Clock SCL High Count Register	0x14	IC_SS_SCL_HCNT	0x0000_0028
Ultra-Fast Speed I2C Clock SCL High Count Register	0x14	IC_UFM_SCL_HCNT	0x0000_0064
Standard Speed I2C Clock SCL Low Count Register	0x18	IC_SS_SCL_LCNT	0x0000_0028
Ultra-Fast Speed I2C Clock SCL Low Count Register	0x18	IC_UFM_SCL_LCNT	0x0000_0008
Fast Mode or Fast Mode Plus I2C Clock SCL High Count Register	0x1C	IC_FS_SCL_HCNT	0x0000_0006
Ultra-Fast Speed mode TBuf Idle Count Register	0x1C	IC_UFM_TBUF_CNT	0x0000_0008
Fast Mode or Fast Mode Plus I2C Clock SCL Low Count Register	0x20	IC_FS_SCL_LCNT	0x0000_000D
High Speed I2C Clock SCL High Count Register	0x24	IC_HS_SCL_HCNT	0x0000_0006
High Speed I2C Clock SCL Low Count Register	0x28	IC_HS_SCL_LCNT	0x0000_0008
I2C Interrupt Status Register	0x2C	IC_INTR_STAT	0x0000_0000

I2C Interrupt Mask Register	0x30	IC_INTR_MASK	0x0000_08FF
I2C Raw Interrupt Status Register	0x34	IC_RAW_INTR_STAT	0x0000_0000
I2C Receive FIFO Threshold Register	0x38	IC_RX_TL	0x0000_0000
I2C Transmit FIFO Threshold Register	0x3C	IC_TX_TL	0x0000_0000
Clear Combined and Individual Interrupt Register	0x40	IC_CLR_INTR	0x0000_0000
Clear RX_UNDER Interrupt Register	0x44	IC_CLR_RX_UNDER	0x0000_0000
Clear RX_OVER Interrupt Register	0x48	IC_CLR_RX_OVER	0x0000_0000
Clear TX_OVER Interrupt Register	0x4C	IC_CLR_TX_OVER	0x0000_0000
Clear RD_REQ Interrupt Register	0x50	IC_CLR_RD_REQ	0x0000_0000
Clear TX_ABRT Interrupt Register	0x54	IC_CLR_TX_ABRT	0x0000_0000
Clear RX_DONE Interrupt Register	0x58	IC_CLR_RX_DONE	0x0000_0000
Clear ACTIVITY Interrupt Register	0x5C	IC_CLR_ACTIVITY	0x0000_0000
Clear STOP_DET Interrupt Register	0x60	IC_CLR_STOP_DET	0x0000_0000
Clear START_DET Interrupt Register	0x64	IC_CLR_START_DET	0x0000_0000
Clear GEN_CALL Interrupt Register	0x68	IC_CLR_GEN_CALL	0x0000_0000
I2C ENABLE Register	0x6C	IC_ENABLE	0x0000_0000
I2C STATUS Register	0x70	IC_STATUS	0x0000_0006
I2C Transmit FIFO Level Register	0x74	IC_TXFLR	0x0000_0000
I2C Receive FIFO Level Register	0x78	IC_RXFLR	0x0000_0000
I2C SDA Hold Time Length Register	0x7C	IC_SDA_HOLD	0x0000_0001
I2C SDA Hold Time Length Register	0x80	IC_SDA_HOLD	0x0000_0000
Generate Slave Data NACK Register	0x84	IC_SLV_DATA_NACK_ONLY	0x0000_0000
DMA Control Register	0x88	IC_DMA_CR	0x0000_0000
DMA Transmit Data Level Register	0x8C	IC_DMA_TDLR	0x0000_0000
DMA Transmit Data Level Register	0x90	IC_DMA_RDLR	0x0000_0000
I2C SDA Setup Register	0x94	IC_SDA_SETUP	0x0000_0064
I2C ACK General Call Register	0x98	IC_ACK_GENERAL_CALL	0x0000_0001
I2C Enable Status Register	0x9C	IC_ENABLE_STATUS	0x0000_0000
I2C SS, FS or FM+ spike suppression limit	0xA0	IC_FS_SPKLEN	0x0000_0000
I2C Ultra-Fast mode spike suppression limit	0xA0	IC_UFM_SPKLEN	0x0000_0001
I2C HS spike suppression limit register	0xA4	IC_HS_SPKLEN	0x0000_0001
Clear RESTART_DET Interrupt Register	0xA8	IC_CLR_RESTART_DET	0x0000_0000
I2C SCL Stuck at Low Timeout register	0xAC	IC_SCL_STUCK_AT_LOW_TIMEOUT	0xFFFF_FFFF
I2C SDA Stuck at Low Timeout register	0xB0	IC_SDA_STUCK_AT_LOW_TIMEOUT	0xFFFF_FFFF

Clear SCL Stuck at Low Detect interrupt Register	0xB4	IC_SDA_STUCK_AT_LOW_TIMEOUT	0x0000_0000
I2C Device-ID Register	0xB8	IC_DEVICE_ID	0x0000_0000
SMBus Slave Clock Extend Timeout register	0xBC	IC_SMBUS_CLK_LOW_SEXT	0xFFFF_FFFF
SMBus Master Clock Extend Timeout register	0xC0	IC_SMBUS_CLK_LOW_MEXT	0xFFFF_FFFF
SMBus Master THigh MAX Bus-idle count Register	0xC4	IC_SMBUS_THIGH_MAX_IDLE_COUNT	0x0000_FFFF
SMBus Interrupt Status Register	0xC8	IC_SMBUS_INTR_STAT	0x0000_0000
SMBus Interrupt Mask Register	0xCC	IC_SMBUS_INTR_MASK	0x0000_07FF
SMBus Raw Interrupt Status Register	0xD0	IC_SMBUS_RAW_INTR_STAT	0x0000_0000
Clear SMBus Interrupt Register	0xD4	IC_CLR_SMBUS_INTR	0x0000_0000
I2C Optional Slave Address Register	0xD8	IC_OPTIONAL_SAR	0x0000_0000
SMBUS ARP UDID LSB Register	0xDC	IC_SMBUS_UDID_LSB	0xFFFF_FFFF
Component Parameter Register 1	0xF4	IC_COMP_PARAM_1	0x0000_0000
I2C Component Version Register	0xF8	IC_COMP_VERSION	0x0000_0000
I2C Component Type Register	0xFC	IC_COMP_TYPE	0x0000_0000

### 15.5.2 I2C Control Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x00 Reset Value = 0x0000\_007D

Name	Bit	Type	Description	Reset Value
RSVD_IC_CON_2	[31:20]	R	IC_CON_2 Reserved bits - Read Only Exists: Always	0x0000
SMBUS_PERSIS-TENT_SLV_AD DR_EN	[19]	R/W	The bit controls to enable DW_apb_i2c slave as persistent or non persistent slave. If the slave is non-PSA then DW_apb_i2c slave device clears the Address valid flag for both General and Directed Reset ARP command else the address valid flag will always set to 1. This bit is applicable only in Slave mode. Values: - 0x1 (ENABLED): SMBus Persistent Slave address control is enabled. - 0x0 (DISABLED): SMBus Persistent Slave address control is disabled. Exists: IC_SMBUS_ARP==1	0x0
SMBUS_ARP_EN	[18]	R/W	This bit controls whether DW_apb_i2c should enable Address Resolution Logic in SMBus Mode. The Slave mode will decode the Address Resolution Protocol commands and respond to it. The DW_apb_i2c slave also includes the generation/validity of PEC byte for Address Resolution Protocol commands. This bit is applicable only in Slave mode. Values: - 0x1 (ENABLED): SMBus ARP control is enabled. - 0x0 (DISABLED): SMBus ARP control is disabled. Exists: IC_SMBUS_ARP==1	0x0
SMBUS_SLAVE_QUICK_EN	[17]	R/W	If this bit is set to 1, DW_apb_i2c slave only receives Quick commands in SMBus Mode. If this bit is set to 0, DW_apb_i2c slave receives all bus protocols but not Quick commands. This bit is applicable only in slave mode. Values: - 0x1 (ENABLED): SMBus SLave is enabled to receive Quick command. - 0x0 (DISABLED): SMBus SLave is disabled to receive Quick command. Exists: IC_SMBUS==1	0x0
OPTIONAL_SAR_CTRL	[16]	R/W	Enables the usage of IC_OPTIONAL_SAR register. If IC_OPTIONAL_SAR ==1, IC_OPTIONAL_SAR value is used as additional slave address. User must program a valid address in IC_OPTIONAL_SAR before writing 1 to this field. If IC_OPTIONAL_SAR ==0, IC_OPTIONAL_SAR value is not used as additional slave address. In this mode only one I2C slave address is used.  Values: - 0x1 (ENABLED): Optional SAR Address Register is enabled. - 0x0 (DISABLED): Optional SAR Address Register is disabled. Exists: IC OPTIONAL SAR==1	0x0

RSVD_IC_CON_1	[15:12]	R	IC_CON_1 Reserved bits - Read Only Exists: Always	0x0000
BUS_CLEAR_FEATURE_CTRL	[11]	R/W	In Master mode: - 1'b1: Bus Clear Feature is enabled. - 1'b0: Bus Clear Feature is Disabled. In Slave mode, this register bit is not applicable.  Values: - 0x1 (ENABLED): Bus Clear Feature is enabled. - 0x0 (DISABLED): Bus Clear Feature is disabled. Exists: IC_BUS_CLEAR_FEATURE==1	0x0
STOP_DET_IF_MASTER_ACTIVE	[10]	R/W	In Master mode: - 1'b1: issues the STOP_DET interrupt only when master is active. - 1'b0: issues the STOP_DET irrespective of whether master is active or not.  Values: - 0x1 (ENABLED): Master issues the STOP_DET interrupt only when master is active - 0x0 (DISABLED): Master issues the STOP_DET interrupt irrespective of whether master is active or not Exists: Always Memory Access: "(IC_STOP_DET_IF_MASTER_ACTIVE==1) ? <i>read-write</i> : <i>read-only</i> "	0x0
RX_FIFO_FULL_HLD_CTRL	[9]	R/W	This bit controls whether DW_apb_i2c should hold the bus when the Rx FIFO is physically full to its RX_BUFFER_DEPTH, as described in the IC_RX_FULL_HLD_BUS_EN parameter.  Values: - 0x1 (ENABLED): Hold bus when RX_FIFO is full - 0x0 (DISABLED): Overflow when RX_FIFO is full Exists: Always Memory Access: "(IC_RX_FULL_HLD_BUS_EN==1) ? <i>read-write</i> : <i>read-only</i> "	0x0
TX_EMPTY_CTRL	[8]	R/W	This bit controls the generation of the TX_EMPTY interrupt, as described in the IC_RAW_INTR_STAT register.  Values: - 0x1 (ENABLED): Controlled generation of TX_EMPTY interrupt - 0x0 (DISABLED): Default behaviour of TX_EMPTY interrupt Exists: Always	0x0
STOP_DET_IFADDRESSED	[7]	R/W	In slave mode: - 1'b1: issues the STOP_DET interrupt only when it is addressed. - 0'b0: issues the STOP_DET irrespective of whether it's addressed or not. NOTE: During a general call address, this slave does not issue the STOP_DET interrupt if STOP_DET_IF_ADDRESSED = 1'b1, even if the slave responds to the general call address by generating ACK. The STOP_DET interrupt is generated only when the transmitted address matches the slave address (SAR). Values: - 0x1 (ENABLED): slave issues STOP_DET intr only if addressed - 0x0 (DISABLED): slave issues STOP_DET intr always Exists: Always	0x0
IC_SLAVE_DISABLE	[6]	R/W	This bit controls whether I2C has its slave disabled, which means once the preset signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE. You have the choice of having the slave enabled or disabled after reset is applied, which means software does not have to configure the slave. By default, the slave is always enabled (in reset state as well). If you need to disable it after reset, set this bit to 1. If this bit is set (slave is disabled), DW_apb_i2c functions only as a master and does not perform any action that requires a slave.  NOTE: Software should ensure that if this bit is written with 0, then bit 0 should also be written with a 0. Values: - 0x1 (SLAVE_DISABLED): Slave mode is disabled - 0x0 (SLAVE_ENABLED): Slave mode is enabled Exists: Always	0x1

IC_RESTART_EN	[5]	R/W	<p>Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several DW_apb_i2c operations.</p> <p>When RESTART is disabled, the master is prohibited from performing the following functions:</p> <ul style="list-style-type: none"> <li>- Sending a START BYTE</li> <li>- Performing any high-speed mode operation</li> <li>- High-speed mode operation</li> <li>- Performing direction changes in combined format mode</li> <li>- Performing a read operation with a 10-bit address</li> </ul> <p>By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple DW_apb_i2c transfers. If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Master restart enabled</li> <li>- 0x0 (DISABLED): Master restart disabled</li> </ul> <p>Exists: Always</p>	0x1
IC_10BITADDR_MASTER	[4]	R/W	<p>If the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to 'No' (0), this bit is named IC_10BITADDR_MASTER and controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. If I2C_DYNAMIC_TAR_UPDATE is set to 'Yes' (1), the function of this bit is handled by bit 12 of IC_TAR register, and becomes a read-only copy called IC_10BITADDR_MASTER_rd_only.</p> <ul style="list-style-type: none"> <li>- 0: 7-bit addressing</li> <li>- 1: 10-bit addressing</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ADDR_10BITS): Master 10Bit addressing mode</li> <li>- 0x0 (ADDR_7BITS): Master 7Bit addressing mode</li> </ul> <p>Exists: I2C_DYNAMIC_TAR_UPDATE == 0</p>	0x1
IC_10BITADDR_SLAVE	[3]	R/W	<p>When acting as a slave, this bit controls whether the DW_apb_i2c responds to 7- or 10-bit addresses.</p> <ul style="list-style-type: none"> <li>- 0: 7-bit addressing. The DW_apb_i2c ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the IC_SAR register are compared.</li> <li>- 1: 10-bit addressing. The DW_apb_i2c responds to only 10-bit addressing transfers that match the full 10 bits of the IC_SAR register.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ADDR_10BITS): Slave 10Bit addressing</li> <li>- 0x0 (ADDR_7BITS): Slave 7Bit addressing</li> </ul> <p>Exists: Always</p>	0x1
SPEED	[2:1]	R/W	<p>These bits control at which speed the DW_apb_i2c operates; its setting is relevant only if one is operating the DW_apb_i2c in master mode. Hardware protects against illegal values being programmed by software. These bits must be programmed appropriately for slave mode also, as it is used to capture correct value of spike filter as per the speed mode.</p> <p>This register should be programmed only with a value in the range of 1 to IC_MAX_SPEED_MODE; otherwise, hardware updates this register with the value of IC_MAX_SPEED_MODE.</p> <p>1: standard mode (100 kbit/s)      2: fast mode (&lt;=400 kbit/s) or fast mode plus (&lt;=1000Kbit/s)      3: high speed mode (3.4 Mbit/s)</p> <p>Note: This field is not applicable when IC_ULTRA_FAST_MODE=1</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (STANDARD): Standard Speed mode of operation</li> <li>- 0x2 (FAST): Fast or Fast Plus mode of operation</li> <li>- 0x3 (HIGH): High Speed mode of operation</li> </ul> <p>Exists: Always</p>	0x2
R/W	[0]	R/W	<p>This bit controls whether the DW_apb_i2c master is enabled.</p> <p>NOTE: Software should ensure that if this bit is written with '1' then bit 6 should also be written with a '1'.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Master mode is enabled</li> <li>- 0x0 (DISABLED): Master mode is disabled</li> </ul> <p>Exists: Always</p>	0x1

### 15.5.3 I2C Target Address Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x04 Reset Value = 0x0000\_1055

Name	Bit	Type	Description	Reset Value
RSVD_IC_TAR_2	[31:17]	R	IC_TAR_2 Reserved bits - Read Only Exists: Always	0x0000
SMBUS_QUICK_CMD	[16]	R/W	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a Quick command is to be performed by the DW_apb_i2c. Values: - 0x1 (ENABLED): Enables programming of QUICK-CMD transmission - 0x0 (DISABLED): Disables programming of QUICK-CMD transmission Exists: IC_SMBUS == 1	0x0
RSVD_IC_TAR_1	[15:14]	R	IC_TAR_1 Reserved bits - Read Only Exists: Always	0x0000
DEVICE_ID	[13]	R/W	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a Device-ID of a particular slave mentioned in IC_TAR[9:0] is to be performed by the DW_apb_i2c Master. - 0: Device-ID is not performed and checks ic_tar[10] to perform either general call or START byte command - 1: Device-ID transfer is performed and bytes based on the number of read commands in the Tx-FIFO are received from the targeted slave and put in the Rx-FIFO.  Values: - 0x1 (ENABLED): Enables programming of DEVICE-ID transmission - 0x0 (DISABLED): Disables programming of DEVICE-ID transmission Exists: IC_DEVICE_ID == 1	0x0
IC_10BITADDR_MASTER	[12]	R/W	This bit controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. - 0: 7-bit addressing - 1: 10-bit addressing  Values: - 0x1 (ADDR_10BITS): Address 10Bit transmission format - 0x0 (ADDR_7BITS): Address 7Bit transmission format Exists: I2C_DYNAMIC_TAR_UPDATE	0x1

SPECIAL	[11]	R/W	<p>This bit indicates whether software performs a Device-ID or General Call or START BYTE command.</p> <ul style="list-style-type: none"> <li>- 0: ignore bit 10 GC_OR_START and use IC_TAR normally</li> <li>- 1: perform special I2C command as specified in Device_ID or GC_OR_START bit</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Enables programming of GENERAL_CALL or START_BYTE transmission</li> <li>- 0x0 (DISABLED): Disables programming of GENERAL_CALL or START_BYTE transmission</li> </ul> <p>Exists: Always</p>	0x0
GC_OR_START	[10]	R/W	<p>If bit 11 (SPECIAL) is set to 1 and bit 13(Device-ID) is set to 0, then this bit indicates whether a General Call or START byte command is to be performed by the DW_apb_i2c.</p> <ul style="list-style-type: none"> <li>- 0: General Call Address - after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The DW_apb_i2c remains in General Call mode until the SPECIAL bit value (bit 11) is cleared.</li> <li>- 1: START BYTE</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (START_BYTE): START byte transmission</li> <li>- 0x0 (GENERAL_CALL): GENERAL_CALL byte transmission</li> </ul> <p>Exists: Always</p>	0x0
IC_TAR	[9:0]	R/W	<p>This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. If the IC_TAR and IC_SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave.</p> <p>Exists: Always</p>	0x55

#### 15.5.4 I2C Slave Address Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0055

Name	Bit	Type	Description	Reset Value
RSVD_IC_SAR	[31:10]	R	IC_SAR Reserved bits - Read Only Exists: Always	0x0000

IC_SAR	[9:0]	R/W	<p>The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>Note: The default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f. The correct operation of the device is not guaranteed if you program the IC_SAR or IC_TAR to a reserved value. Refer to Table "I2C/SMBus Definition of Bits in First Byte" for a complete list of these reserved values.</p> <p>Exists: Always</p>	0x55
--------	-------	-----	--	------

### 15.5.5 I2C High Speed Master Mode Code Address Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD_IC_HS_MAR	[31:3]	R	IC_HS_MAR Reserved bits - Read Only Exists: Always	0x0000
IC_HS_MAR	[2:0]	R/W	<p>This bit field holds the value of the I2C HS mode master code. HS-mode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight high-speed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7. This register goes away and becomes read-only returning 0's if the IC_MAX_SPEED_MODE configuration parameter is set to either Standard (1) or Fast (2).</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>Exists: Always</p>	0x1

### 15.5.6 I2C Rx/Tx Data Buffer and Command Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_DATA_CMD	[31:12]	R	IC_DATA_CMD Reserved bits - Read Only Exists: Always Volatile: true	0x0000
FIRST_DATA_BYTE	[11]	R	Indicates the first data byte received after the address phase for receive transfer in Master receiver or Slave receiver mode.  NOTE: In case of APB_DATA_WIDTH=8, 1. The user has to perform two APB Reads to IC_DATA_CMD in order to get status on 11 bit. 2. Inorder to read the 11 bit, the user has to perform the first data byte read [7:0] (offset 0x10) and then perform the second read[15:8](offset 0x11) in order to know the status of 11 bit (whether the data received in previous read is a first data byte or not). 3. The 11th bit is an optional read field, user can ignore 2nd byte read [15:8] (offset 0x11) if not interested in FIRST_DATA_BYTE status. Values: - 0x1 (ACTIVE): Non sequential data byte received - 0x0 (INACTIVE): Sequential data byte received Exists: IC_FIRST_DATA_BYTE_STATUS == 1 Volatile: true	0x0
RESTART	[10]	W	This bit controls whether a RESTART is issued before the byte is sent or received. This bit is available only if IC_EMPTYFIFO_HOLD_MASTER_EN is configured to 1. 1 - If IC_RESTART_EN is 1, a RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command; if IC_RESTART_EN is 0, a STOP followed by a START is issued instead. 0 - If IC_RESTART_EN is 1, a RESTART is issued only if the transfer direction is changing from the previous command; if IC_RESTART_EN is 0, a STOP followed by a START is issued instead.  Values: - 0x1 (ENABLE): Issue RESTART before this command - 0x0 (DISABLE): Donot Issue RESTART before this command Exists: IC_EMPTYFIFO_HOLD_MASTER_EN Volatile: true	0x0

STOP	[9]	W	<p>This bit controls whether a STOP is issued after the byte is sent or received. This bit is available only if IC_EMPTYFIFO_HOLD_MASTER_EN is configured to 1.</p> <ul style="list-style-type: none"> <li>- 1 - STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus.</li> <li>- 0 - STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLE): Issue STOP after this command</li> <li>- 0x0 (DISABLE): Donot Issue STOP after this command</li> </ul> <p>Exists: IC_EMPTYFIFO_HOLD_MASTER_EN Volatile: true</p>	0x0
CMD	[8]	W	<p>This bit controls whether a read or a write is performed. This bit does not control the direction when the DW_apb_i2c acts as a slave. It controls only the direction when it acts as a master. When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a "don't care" because writes to this register are not required. In slave-transmitter mode, a "0" indicates that the data in IC_DATA_CMD is to be transmitted.</p> <p>When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR register has been cleared. If a "1" is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (READ): Master Read Command</li> <li>- 0x0 (WRITE): Master Write Command</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
DAT	[7:0]	R/W	<p>This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the DW_apb_i2c. However, when you read this register, these bits return the value of data received on the DW_apb_i2c interface.</p> <p>Exists: Always Volatile: true</p>	0x0

### 15.5.7 Standard Speed I2C Clock SCL High Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)

- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x14 Reset Value = 0x0000\_0028

Name	Bit	Type	Description	Reset Value
RSVD_IC_SS_SCL_HIGH_COU NT	[31:16]	R	IC_SS_SCL_HCNT Reserved bits - Read Only Exists: Always	0x0000
IC_SS_SCL_HCNT	[15:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. For more information, refer to "IC_CLK Frequency Configuration".</p> <p>This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>NOTE: This register must not be programmed to a value higher than 65525, because DW_apb_i2c uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.</p> <p>Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? <i>read-only</i> : <i>read-write</i>"</p>	0x28

### 15.5.8 Ultra-Fast Speed I2C Clock SCL High Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x14 Reset Value = 0x0000\_0064

Name	Bit	Type	Description	Reset Value
RSVD_IC_UFM_SCL_HCNT	[31:16]	R	IC_UFM_SCL_HCNT Reserved bits - Read Only Exists: Always	0x0000

IC_UFM_SCL_HCNT	[15:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for Ultra-Fast speed. For more information, refer to "IC_CLK Frequency Configuration".</p> <p>This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 3; hardware prevents values less than this being written, and if attempted results in 3 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? read-only : read-write"</p>	0x64
-----------------	--------	-----	--	------

### 15.5.9 Standard Speed I2C Clock SCL Low Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x18 Reset Value = 0x0000\_0028

Name	Bit	Type	Description	Reset Value
RSVD_IC_SS_SCL_LOW_COUNT	[31:16]	R	RSVD_IC_SS_SCL_LOW_COUNT Reserved bits - Read Only Exists: Always	0x0000
* Varies	[15:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. For more information, refer to "IC_CLK Frequency Configuration".</p> <p>This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of DW_apb_i2c. The lower byte must be programmed first, and then the upper byte is programmed.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? read-only : read-write"</p>	0x28

### 15.5.10 Ultra-Fast Speed I2C Clock SCL Low Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x18 Reset Value = 0x0000\_0008

Name	Bit	Type	Description	Reset Value
RSVD_IC_UFM_SCL_LCNT	[31:16]	R	IC_UFM_SCL_LCNT Reserved bits - Read Only Exists: Always	0x0000
IC_UFM_SCL_LCNT	[15:0]	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for Ultra-Fast speed. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect. The minimum valid value is 5; hardware prevents values less than this being written, and if attempted, results in 5 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of DW_apb_i2c. The lower byte must be programmed first, and then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? ‐read-only‐ : ‐read-write‐"	0x8

### 15.5.11 Fast Mode or Fast Mode Plus I2C Clock SCL High Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)

- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x1C Reset Value = 0x0000\_0006

Name	Bit	Type	Description	Reset Value
RSVD_IC_FS_SCL_HCNT	[31:16]	R	IC_FS_SCL_HCNT Reserved bits - Read Only Exists: Always	0x0000
IC_FS_SCL_HCNT	[15:0]	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast mode or fast mode plus. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to "IC_CLK Frequency Configuration".  This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH == 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed.  Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? read-only" : "read-write"	0x6

### 15.5.12 Ultra-Fast Speed mode TBuf Idle Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x1C Reset Value = 0x0000\_0008

Name	Bit	Type	Description	Reset Value
RSVD_IC_UFM_TBUF_CNT	[31:16]	R	IC_UFM_TBUF_CNT Reserved bits - Read Only Exists: Always	0x0000

IC_UFM_TBUF_CNT	[15:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the Bus-Free time between a STOP and STOP condition count for Ultra-Fast speed.</p> <p>This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first and then the upper byte is programmed. When the configuration parameter.</p> <p>NOTE: The DW_apb_i2c will add 9 ic_clks after tBuf time is expired to generate START on the Bus.</p> <p>Exists: Always</p>	0x8
-----------------	--------	-----	---	-----

### 15.5.13 Fast Mode or Fast Mode Plus I2C Clock SCL Low Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x20 Reset Value = 0x0000\_000D

Name	Bit	Type	Description	Reset Value
RSVD_IC_FS_SCL_LCNT	[31:16]	R	IC_FS_SCL_LCNT Reserved bits - Read Only Exists: Always	0x0000
IC_FS_SCL_LCNT	[15:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast speed. It is used in high- speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to "IC_CLK Frequency Configuration".</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. If the value is less than 8 then the count value gets changed to 8.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? read-only: read-write"</p>	0xd

### 15.5.14 High Speed I2C Clock SCL High Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x24 Reset Value = 0x0000\_0006

Name	Bit	Type	Description	Reset Value
RSVD_IC_HS_SCL_HCNT	[31:16]	R	IC_HS_SCL_HCNT Reserved bits - Read Only Exists: Always	0x0000
IC_HS_SCL_HCNT	[15:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high period count for high speed.refer to "IC_CLK Frequency Configuration".</p> <p>The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns. This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE != high.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed.</p> <p>Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? read-only": read-write"</p>	0x6

### 15.5.15 High Speed I2C Clock SCL Low Count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)

- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x28 Reset Value = 0x0000\_0008

Name	Bit	Type	Description	Reset Value
RSVD_IC_HS_SCL_LOW_CNT	[31:16]	R	IC_HS_SCL_LCNT Reserved bits - Read Only Exists: Always	0x0000
IC_HS_SCL_LCNT	[15:0]	R/W	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for high speed. For more information, refer to "IC_CLK Frequency Configuration". The SCL low time depends on the loading of the bus. For 100pF loading, the SCL low time is 160ns; for 400pF loading, the SCL low time is 320ns. This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE != high. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with APB_DATA_WIDTH == 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. If the value is less than 8 then the count value gets changed to 8.  Exists: Always Memory Access: "(IC_HC_COUNT_VALUES==1) ? ‐read-only‐ : ‐read-write‐"	0x8

### 15.5.16 I2C Interrupt Status Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x2C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value

RSVD_IC_INTR_STAT	[31:15]	R	IC_INTR_STAT Reserved bits - Read Only Exists: Always Volatile: true	0x0000
R_SCL_STUCK_AT_LOW	[14]	R	See IC_RAW_INTR_STAT for a detailed description of R_SCL_STUCK_AT_LOW bit. Values: - 0x1 (ACTIVE): R_SCL_STUCK_AT_LOW interrupt is active - 0x0 (INACTIVE): R_SCL_STUCK_AT_LOW interrupt is inactive Exists: IC_BUS_CLEAR_FEATURE==1 Volatile: true	0x0
R_MASTER_ON_HOLD	[13]	R	See IC_RAW_INTR_STAT for a detailed description of R_MASTER_ON_HOLD bit.  Values: - 0x1 (ACTIVE): R_MASTER_ON_HOLD interrupt is active - 0x0 (INACTIVE): R_MASTER_ON_HOLD interrupt is inactive Exists: Always Volatile: true	0x0
R_RESTART_DET	[12]	R	See IC_RAW_INTR_STAT for a detailed description of R_RESTART_DET bit.  Values: - 0x1 (ACTIVE): R_RESTART_DET interrupt is active - 0x0 (INACTIVE): R_RESTART_DET interrupt is inactive Exists: Always Volatile: true	0x0
R_GEN_CALL	[11]	R	See IC_RAW_INTR_STAT for a detailed description of R_GEN_CALL bit.  Values: - 0x1 (ACTIVE): R_GEN_CALL interrupt is active - 0x0 (INACTIVE): R_GEN_CALL interrupt is inactive Exists: Always Volatile: true	0x0
R_START_DET	[10]	R	See IC_RAW_INTR_STAT for a detailed description of R_START_DET bit.  Values: - 0x1 (ACTIVE): R_START_DET interrupt is active - 0x0 (INACTIVE): R_START_DET interrupt is inactive Exists: Always Volatile: true	0x0
R_STOP_DET	[9]	R	See IC_RAW_INTR_STAT for a detailed description of R_STOP_DET bit.  Values: - 0x1 (ACTIVE): R_STOP_DET interrupt is active - 0x0 (INACTIVE): R_STOP_DET interrupt is inactive Exists: Always Volatile: true	0x0
R_ACTIVITY	[8]	R	See IC_RAW_INTR_STAT for a detailed description of R_ACTIVITY bit.  Values: - 0x1 (ACTIVE): R_ACTIVITY interrupt is active - 0x0 (INACTIVE): R_ACTIVITY interrupt is inactive Exists: Always Volatile: true	0x0
R_RX_DONE	[7]	R	See IC_RAW_INTR_STAT for a detailed description of R_RX_DONE bit.  Values: - 0x1 (ACTIVE): R_RX_DONE interrupt is active - 0x0 (INACTIVE): R_RX_DONE interrupt is inactive Exists: IC_ULTRA_FAST_MODE==0 Volatile: true	0x0
R_TX_ABRT	[6]	R	See IC_RAW_INTR_STAT for a detailed description of R_TX_ABRT bit.  Values: - 0x1 (ACTIVE): R_TX_ABRT interrupt is active - 0x0 (INACTIVE): R_TX_ABRT interrupt is inactive Exists: Always Volatile: true	0x0

R_RD_REQ	[5]	R	<p>See IC_RAW_INTR_STAT for a detailed description of R_RD_REQ bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): R_RD_REQ interrupt is active</li> <li>- 0x0 (INACTIVE): R_RD_REQ interrupt is inactive</li> </ul> <p>Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
R_TX_EMPTY	[4]	R	<p>See IC_RAW_INTR_STAT for a detailed description of R_TX_EMPTY bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): R_TX_EMPTY interrupt is active</li> <li>- 0x0 (INACTIVE): R_TX_EMPTY interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
R_TX_OVER	[3]	R	<p>See IC_RAW_INTR_STAT for a detailed description of R_TX_OVER bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): R_TX_OVER interrupt is active</li> <li>- 0x0 (INACTIVE): R_TX_OVER interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
R_RX_FULL	[2]	R	<p>See IC_RAW_INTR_STAT for a detailed description of R_RX_FULL bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): R_RX_FULL interrupt is active</li> <li>- 0x0 (INACTIVE): R_RX_FULL interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
R_RX_OVER	[1]	R	<p>See IC_RAW_INTR_STAT for a detailed description of R_RX_OVER bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): R_RX_OVER interrupt is active</li> <li>- 0x0 (INACTIVE): R_RX_OVER interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
R_RX_UNDER	[0]	R	<p>See IC_RAW_INTR_STAT for a detailed description of R_RX_UNDER bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): RX_UNDER interrupt is active</li> <li>- 0x0 (INACTIVE): RX_UNDER interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

### 15.5.17 I2C Interrupt Mask Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)

- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x30 Reset Value = 0x0000\_08FF

Name	Bit	Type	Description	Reset Value
RSVD_IC_INTR_STAT	[31:15]	R	IC_INTR_STAT Reserved bits - Read Only Exists: Always	0x0000
M_SCL_STUCK_AT_LOW	[14]	R/W	This bit masks the R_SCL_STUCK_AT_LOW interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): SCL_STUCK_AT_LOW interrupt is unmasked - 0x0 (ENABLED): SCL_STUCK_AT_LOW interrupt is masked Exists: IC_BUS_CLEAR_FEATURE==1	0x0
M_MASTER_ON_HOLD	[13]	R/W	This bit masks the R_MASTER_ON_HOLD interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): MASTER_ON_HOLD interrupt is unmasked - 0x0 (ENABLED): MASTER_ON_HOLD interrupt is masked Exists: I2C_DYNAMIC_TAR_UPDATE == 1 & IC_EMPTYFIFO_HOLD_MASTER_EN == 1	0x0
M_RESTART_DET	[12]	R/W	This bit masks the R_RESTART_DET interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): RESTART_DET interrupt is unmasked - 0x0 (ENABLED): RESTART_DET interrupt is masked Exists: IC_SLV_RESTART_DET_EN == 1	0x0
M_GEN_CALL	[11]	R/W	This bit masks the R_GEN_CALL interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): GEN_CALL interrupt is unmasked - 0x0 (ENABLED): GEN_CALL interrupt is masked Exists: Always	0x1
M_START_DET	[10]	R/W	This bit masks the R_START_DET interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): START_DET interrupt is unmasked - 0x0 (ENABLED): START_DET interrupt is masked Exists: Always	0x0
M_STOP_DET	[9]	R/W	This bit masks the R_STOP_DET interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): STOP_DET interrupt is unmasked - 0x0 (ENABLED): STOP_DET interrupt is masked Exists: Always	0x0
M_ACTIVITY	[8]	R/W	This bit masks the R_ACTIVITY interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): ACTIVITY interrupt is unmasked - 0x0 (ENABLED): ACTIVITY interrupt is masked Exists: Always	0x0
M_RX_DONE	[7]	R/W	This bit masks the R_RX_DONE interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): RX_DONE interrupt is unmasked - 0x0 (ENABLED): RX_DONE interrupt is masked Exists: IC_ULTRA_FAST_MODE==0	0x1
M_TX_ABRT	[6]	R/W	This bit masks the R_TX_ABRT interrupt in IC_INTR_STAT register.  Values: - 0x1 (DISABLED): TX_ABORT interrupt is unmasked - 0x0 (ENABLED): TX_ABORT interrupt is masked Exists: Always	0x1

M_RD_REQ	[5]	R/W	<p>This bit masks the R_RD_REQ interrupt in IC_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): RD_REQ interrupt is unmasked</li> <li>- 0x0 (ENABLED): RD_REQ interrupt is masked</li> </ul> <p>Exists: IC_ULTRA_FAST_MODE==0</p>	0x1
M_TX_EMPTY	[4]	R/W	<p>This bit masks the R_TX_EMPTY interrupt in IC_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): TX_EMPTY interrupt is unmasked</li> <li>- 0x0 (ENABLED): TX_EMPTY interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1
M_TX_OVER	[3]	R/W	<p>This bit masks the R_TX_OVER interrupt in IC_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): TX_OVER interrupt is unmasked</li> <li>- 0x0 (ENABLED): TX_OVER interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1
M_RX_FULL	[2]	R/W	<p>This bit masks the R_RX_FULL interrupt in IC_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): RX_FULL interrupt is unmasked</li> <li>- 0x0 (ENABLED): RX_FULL interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1
M_RX_OVER	[1]	R/W	<p>This bit masks the R_RX_OVER interrupt in IC_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): RX_OVER interrupt is unmasked</li> <li>- 0x0 (ENABLED): RX_OVER interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1
M_RX_UNDER	[0]	R/W	<p>This bit masks the R_RX_UNDER interrupt in IC_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): RX_UNDER interrupt is unmasked</li> <li>- 0x0 (ENABLED): RX_UNDER interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1

### 15.5.18 I2C Raw Interrupt Status Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_RAW_INTR_STAT	[31:15]	R	IC_RAW_INTR_STAT Reserved bits - Read Only Exists: Always Volatile: true	0x0000
SCL_STUCK_AT_LOW	[14]	R	Indicates whether the SCL Line is stuck at low for the IC_SCL_STUCK_LOW_TIMEOUT number of ic_clk periods. Enabled only when IC_BUS_CLEAR_FEATURE=1 and IC_ULTRA_FAST_MODE=0.  Values: - 0x1 (ACTIVE): SCL_STUCK_AT_LOW interrupt is active - 0x0 (INACTIVE): SCL_STUCK_AT_LOW interrupt is inactive. Exists: IC_BUS_CLEAR_FEATURE==1 Volatile: true	0x0
MASTER_ON_HOLD	[13]	R	Indicates whether master is holding the bus and TX FIFO is empty. Enabled only when I2C_DYNAMIC_TAR_UPDATE=1 and IC_EMPTYFIFO_HOLD_MASTER_EN=1.  Values: - 0x1 (ACTIVE): MASTER_ON_HOLD interrupt is active - 0x0 (INACTIVE): MASTER_ON_HOLD interrupt is inactive Exists: Always Volatile: true	0x0
RESTART_DET	[12]	R	Indicates whether a RESTART condition has occurred on the I2C interface when DW_apb_i2c is operating in Slave mode and the slave is being addressed. Enabled only when IC_SLV_RESTART_DET_EN=1. Note: However, in high-speed mode or during a START BYTE transfer, the RESTART comes before the address field as per the I2C protocol. In this case, the slave is not the addressed slave when the RESTART is issued, therefore DW_apb_i2c does not generate the RESTART_DET interrupt.  Values: - 0x1 (ACTIVE): RESTART_DET interrupt is active - 0x0 (INACTIVE): RESTART_DET interrupt is inactive Exists: Always Volatile: true	0x0
GEN_CALL	[11]	R	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling DW_apb_i2c or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. DW_apb_i2c stores the received data in the Rx buffer.  Values: - 0x1 (ACTIVE): GEN_CALL interrupt is active - 0x0 (INACTIVE): GEN_CALL interrupt is inactive Exists: Always Volatile: true	0x0
START_DET	[10]	R	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode.  Values: - 0x1 (ACTIVE): START_DET interrupt is active - 0x0 (INACTIVE): START_DET interrupt is inactive Exists: Always Volatile: true	0x0

STOP_DET	[9]	R	<p>Indicates whether a STOP condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode.</p> <p>In Slave Mode:</p> <ul style="list-style-type: none"> <li>- If IC_CON[7]=1'b1 (STOP_DET_IFADDRESSED), the STOP_DET interrupt will be issued only if slave is addressed.</li> </ul> <p>Note: During a general call address, this slave does not issue a STOP_DET interrupt if STOP_DET_IF_ADDRESSED=1'b1, even if the slave responds to the general call address by generating ACK. The STOP_DET interrupt is generated only when the transmitted address matches the slave address (SAR).</p> <ul style="list-style-type: none"> <li>- If IC_CON[7]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is issued irrespective of whether it is being addressed.</li> </ul> <p>In Master Mode:</p> <ul style="list-style-type: none"> <li>- If IC_CON[10]=1'b1 (STOP_DET_IF_MASTER_ACTIVE), the STOP_DET interrupt will be issued only if Master is active.</li> <li>- If IC_CON[10]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt will be issued irrespective of whether master is active or not.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): STOP_DET interrupt is active</li> <li>- 0x0 (INACTIVE): STOP_DET interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
ACTIVITY	[8]	R	<p>This bit captures DW_apb_i2c activity and stays set until it is cleared. There are four ways to clear it:</p> <ul style="list-style-type: none"> <li>- Disabling the DW_apb_i2c</li> <li>- Reading the IC_CLR_ACTIVITY register</li> <li>- Reading the IC_CLR_INTR register</li> <li>- System reset</li> </ul> <p>Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the DW_apb_i2c module is idle, this bit remains set until cleared, indicating that there was activity on the bus.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): RAW_INTR_ACTIVITY interrupt is active</li> <li>- 0x0 (INACTIVE): RAW_INTR_ACTIVITY interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
RX_DONE	[7]	R	<p>When the DW_apb_i2c is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): RX_DONE interrupt is active</li> <li>- 0x0 (INACTIVE): RX_DONE interrupt is inactive</li> </ul> <p>Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
TX_ABRT	[6]	R	<p>This bit indicates if DW_apb_i2c, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places.</p> <p>Note: The DW_apb_i2c flushes/resets/empties only the TX_FIFO whenever there is a transmit abort caused by any of the events tracked by the IC_TX_ABRT_SOURCE register. The Tx FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the Tx FIFO is then ready to accept more data bytes from the APB interface. RX FIFO flush because of TX_ABRT is controlled by the coreConsultant parameter IC_AVOID_RX_FIFO_FLUSH_ON_TX_ABRT.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): TX_ABRT interrupt is active</li> <li>- 0x0 (INACTIVE): TX_ABRT interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

RD_REQ	[5]	R	<p>This bit is set to 1 when DW_apb_I2c is acting as a slave and another I2C master is attempting to read data from DW_apb_I2c. The DW_apb_I2c holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): RD_REQ interrupt is active</li> <li>- 0x0 (INACTIVE): RD_REQ interrupt is inactive</li> </ul> <p>Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
TX_EMPTY	[4]	R	<p>The behavior of the TX_EMPTY interrupt status differs based on the TX_EMPTY_CTRL selection in the IC_CON register.</p> <ul style="list-style-type: none"> <li>- When TX_EMPTY_CTRL = 0: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register.</li> <li>- When TX_EMPTY_CTRL = 1: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register and the transmission of the address/data from the internal shift register for the most recently popped command is completed.</li> </ul> <p>It is automatically cleared by hardware when the buffer level goes above the threshold. When IC_ENABLE[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer any activity, then with ic_en=0, this bit is set to 0.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): TX_EMPTY interrupt is active</li> <li>- 0x0 (INACTIVE): TX_EMPTY interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
TX_OVER	[3]	R	<p>Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): TX_OVER interrupt is active</li> <li>- 0x0 (INACTIVE): TX_OVER interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
RX_FULL	[2]	R	<p>Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): RX_FULL interrupt is active</li> <li>- 0x0 (INACTIVE): RX_FULL interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

RX_OVER	[1]	R	<p>Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_I2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> <p>Note: If the configuration parameter IC_RX_FULL_HLD_BUS_EN is enabled and bit 9 of the IC_CON register (RX_FIFO_FULL_HLD_CTRL) is programmed to HIGH, then the RX_OVER interrupt never occurs, because the Rx FIFO never overflows.</p> <p>Values:            - 0x1 (ACTIVE): RX_OVER interrupt is active            - 0x0 (INACTIVE): RX_OVER interrupt is inactive</p> <p>Exists: Always            Volatile: true</p>	0x0
RX_UNDER	[0]	R	<p>Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> <p>Values:            - 0x1 (ACTIVE): RX_UNDER interrupt is active            - 0x0 (INACTIVE): RX_UNDER interrupt is inactive</p> <p>Exists: Always            Volatile: true</p>	0x0

### 15.5.19 I2C Receive FIFO Threshold Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_RX_TL	[31:8]	R	IC_RX_TL Reserved bits - Read Only Exists: Always	0x0000
RX_TL	[7:0]	R/W	<p>Receive FIFO Threshold Level. Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.</p> <p>Exists: Always</p>	0x0

### 15.5.20 I2C Transmit FIFO Threshold Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x3C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_TX_TL	[31:8]	R	IC_TX_TL Reserved bits - Read Only Exists: Always	0x0000
TX_TL	[7:0]	R/W	Transmit FIFO Threshold Level. Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries.  Exists: Always	0x0

### 15.5.21 Clear Combined and Individual Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_INTR	[31:1]	R	CLR_INTR Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_INTR	[0]	R	Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. Exists: Always Volatile: true	0x0

### 15.5.22 Clear RX\_UNDER Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_RX_UNDER	[31:1]	R	IC_CLR_RX_UNDER Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_RX_UNDER	[0]	R	Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.23 Clear RX\_OVER Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)

- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x48 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_RX_OVER	[31:1]	R	IC_CLR_RX_OVER Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_RX_OVER	[0]	R	Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.24 Clear TX\_OVER Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x4C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_TX_OVER	[31:1]	R	IC_CLR_TX_OVER Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_TX_OVER	[0]	R	Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.25 Clear RD\_REQ Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_RD_REQ	[31:1]	R	IC_CLR_RD_REQ Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_RD_REQ	[0]	R	Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.26 Clear TX\_ABRT Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x54 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

RSVD_IC_CLR_TX_ABRT	[31:1]	R	IC_CLR_TX_ABRT Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_TX_ABRT	[0]	R	Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. Exists: Always Volatile: true	0x0

### 15.5.27 Clear RX\_DONE Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x58 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_RX_DONE	[31:1]	R	IC_CLR_RX_DONE Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_RX_DONE	[0]	R	Read this register to clear the RX_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.28 Clear ACTIVITY Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)

- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x5C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_ACTIVITY	[31:1]	R	IC_CLR_ACTIVITY Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_ACTIVITY	[0]	R	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.29 Clear STOP\_DET Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x60 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_STOP_DET	[31:1]	R	IC_CLR_STOP_DET Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_STOP_DET	[0]	R	Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.30 Clear START\_DET Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x64 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_START_DET	[31:1]	R	IC_CLR_START_DET Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_START_DET	[0]	R	Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.31 Clear GEN\_CALL Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x68 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

RSVD_IC_CLR_GEN_CALL	[31:1]	R	IC_CLR_GEN_CALL Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_GEN_CALL	[0]	R	Read this register to clear the GEN_CALL interrupt (bit 11) of IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.32 I2C ENABLE Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x6C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_ENABLE_2	[31:19]	R	IC_ENABLE Reserved bits - Read Only Exists: Always	0x0000
SMBUS_ALERT_EN	[18]	R/W	The SMBUS_ALERT_CTRL register bit is used to control assertion of SMBALERT signal. - 1: Assert SMBALERT signal This register bit is auto-cleared after detection of Acknowledgement from master for Alert Response address.. Values: - 0x1 (ALERT_ENABLED): Slave initiates the Alert signal to indicate SMBus Host - 0x0 (SUSPEND_DISABLED): Slave will not initiates the Alert signal to indicate SMBus Host. Exists: IC_SMBUS_SUSPEND_ALERT==1	0x0
SMBUS_SUSPEND_EN	[17]	R/W	The SMBUS_SUSPEND_EN register bit is used to control assertion and de-assertion of SMBSUS signal. - 0: De-assert SMBSUS signal - 1: Assert SMBSUS signal Values: - 0x1 (ENABLED): Host/Master initiates the SMBUS system to enter Suspend Mode. - 0x0 (DISABLED): Host/Master will not initiates the SMBUS system to enter Suspend Mode. Exists: IC_SMBUS_SUSPEND_ALERT==1	0x0

SMBUS_CLK_RESET	[16]	R/W	<p>This bit is used in SMBus Host mode to initiate the SMBus Master Clock Reset. This bit should be enabled only when Master is in idle. Whenever this bit is enabled, the SMBCLK is held low for the IC_SCL_STUCK_TIMEOUT ic_clk cycles to reset the SMBus slave devices.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Master initiates the SMBUS Clock Reset Mechanism.</li> <li>- 0x0 (DISABLED): Master will not initiates SMBUS Clock Reset Mechanism.</li> </ul> <p>Exists: IC_SMBUS==1</p>	0x0
RSVD_IC_ENABLE_1	[15:4]	R	<p>RSVD_IC_ENABLE_1 Reserved bits - Read Only</p> <p>Exists: Always</p>	0x0000
SDA_STUCK_RECOVERY_ENA_BLE	[3]	R/W	<p>If SDA is stuck at low indicated through the TX_ABORT interrupt (IC_TX_ABRT_SOURCE[17]), then this bit is used as a control knob to initiate the SDA Recovery Mechanism (that is, send at most 9 SCL clocks and STOP to release the SDA line) and then this bit gets auto clear.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (SDA_STUCK_RECOVERY_ENABLED): Master initiates the SDA stuck at low recovery mechanism.</li> <li>- 0x0 (SDA_STUCK_RECOVERY_DISABLED): Master disabled the SDA stuck at low recovery mechanism.</li> </ul> <p>Exists: IC_BUS_CLEAR_FEATURE==1</p>	0x0
TX_CMD_BLOCK	[2]	R/W	<p>In Master mode:</p> <ul style="list-style-type: none"> <li>- 1'b1: Blocks the transmission of data on I2C bus even if Tx FIFO has data to transmit.</li> <li>- 1'b0: The transmission of data starts on I2C bus automatically, as soon as the first data is available in the Tx FIFO.</li> </ul> <p>Note: To block the execution of Master commands, set the TX_CMD_BLOCK bit only when Tx FIFO is empty (IC_STATUS[2]==1) and Master is in Idle state (IC_STATUS[5] == 0). Any further commands put in the Tx FIFO are not executed until TX_CMD_BLOCK bit is unset.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (BLOCKED): Tx Command execution blocked</li> <li>- 0x0 (NOT_BLOCKED): Tx Command execution not blocked</li> </ul> <p>Exists: Always</p>	0x0
ABORT	[1]	R/W	<p>When set, the controller initiates the transfer abort.</p> <ul style="list-style-type: none"> <li>- 0: ABORT not initiated or ABORT done</li> <li>- 1: ABORT operation in progress</li> </ul> <p>The software can abort the I2C transfer in master mode by setting this bit. The software can set this bit only when ENABLE is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set. In response to an ABORT, the controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABORT interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation.</p> <p>For a detailed description on how to abort I2C transfers, refer to "Aborting I2C Transfers".</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): ABORT operation in progress</li> <li>- 0x0 (DISABLE): ABORT operation not in progress</li> </ul> <p>Exists: Always</p>	0x0

ENABLE	[0]	R/W	<p>Controls whether the DW_apb_i2c is enabled.</p> <ul style="list-style-type: none"> <li>- 0: Disables DW_apb_i2c (TX and RX FIFOs are held in an erased state)</li> <li>- 1: Enables DW_apb_i2c</li> </ul> <p>Software can disable DW_apb_i2c while it is active. However, it is important that care be taken to ensure that DW_apb_i2c is disabled properly. A recommended procedure is described in "Disabling DW_apb_i2c".</p> <p>When DW_apb_i2c is disabled, the following occurs:</p> <ul style="list-style-type: none"> <li>- The TX FIFO and RX FIFO get flushed.</li> <li>- Status bits in the IC_INTR_STAT register are still active until DW_apb_i2c goes into IDLE state.</li> </ul> <p>If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the DW_apb_i2c stops the current transfer at the end of the current byte and does not acknowledge the transfer.</p> <p>In systems with asynchronous pclk and ic_clk when IC_CLK_TYPE parameter set to asynchronous (1), there is a two ic_clk delay when enabling or disabling the DW_apb_i2c. For a detailed description on how to disable DW_apb_i2c, refer to "Disabling DW_apb_i2c"</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): I2C is enabled</li> <li>- 0x0 (DISABLED): I2C is disabled</li> </ul> <p>Exists: Always</p>	0x0
--------	-----	-----	--	-----

### 15.5.33 I2C STATUS Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x70 Reset Value = 0x0000\_0006

Name	Bit	Type	Description	Reset Value
RSVD_IC_STATUS_2	[31:21]	R	IC_STATUS Reserved bits - Read Only Exists: Always Volatile: true	0x0000
SMBUS_ALERT_STATUS	[20]	R	<p>This bit indicates the status of the SMBus Alert signal (ic_smabalert_in_n). This signal is asserted when the SMBus Alert signal is asserted by the SMBus Device.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SMBUS Alert is asserted.</li> <li>- 0x0 (INACTIVE): SMBUS Alert is not asserted. Exists: IC_SMBUS_SUSPEND_ALERT==1 Volatile: true</li> </ul>	0x0

SMBUS_SUSPEND_STATUS	[19]	R	<p>This bit indicates the status of the SMBus Suspend signal (ic_smbus_in_n). This signal is asserted when the SMBus Suspend signal is asserted by the SMBus Host.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SMBUS System is in Suspended mode.</li> <li>- 0x0 (INACTIVE): SMBUS System is not in Suspended mode.</li> </ul> <p>Exists: IC_SMBUS_SUSPEND_ALERT==1 Volatile: true</p>	0x0
SMBUS_SLAVE_ADDR_RESOLVED	[18]	R	<p>This bit indicates whether the slave address (ic_sar) is resolved by the ARP Master.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SMBUS Slave Address is Resolved.</li> <li>- 0x0 (INACTIVE): SMBUS Slave Address is not Resolved.</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
SMBUS_SLAVE_ADDR_VALID	[17]	R	<p>This bit indicates whether the slave address (ic_sar) is valid or not.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SMBUS Slave Address is Valid.</li> <li>- 0x0 (INACTIVE): SMBUS Slave Address is not valid.</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
SMBUS_QUICK_CMD_BIT	[16]	R	<p>This bit indicates the R/W bit of the Quick command received. This bit will be cleared after the user has read this bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SMBUS QUICK CMD Read/write is set to 1.</li> <li>- 0x0 (INACTIVE): SMBUS QUICK CMD Read/write is set to 0.</li> </ul> <p>Exists: IC_SMBUS==1 Volatile: true</p>	0x0
RSVD_IC_STATUS_1	[15:12]	R	<p>RSVD_IC_STATUS_1 Reserved bits - Read Only</p> <p>Exists: Always Volatile: true</p>	0x0000
SDA_STUCK_NOT_RECOVERED	[11]	R	<p>This bit indicates that SDA stuck at low is not recovered after the recovery mechanism. In Slave mode, this register bit is not applicable.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SDA Stuck at low is recovered after recovery mechanism.</li> <li>- 0x0 (INACTIVE): SDA Stuck at low is not recovered after recovery mechanism.</li> </ul> <p>Exists: IC_BUS_CLEAR_FEATURE==1 Volatile: true</p>	0x0
SLV_HOLD_RX_FIFO_FULL	[10]	R	<p>This bit indicates the BUS Hold in Slave mode due to Rx FIFO is Full and an additional byte has been received (This kind of Bus hold is applicable if IC_RX_FULL_HLD_BUS_EN is set to 1).</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Slave holds the bus due to Rx FIFO is full</li> <li>- 0x0 (INACTIVE): Slave is not holding the bus or Bus hold is not due to Rx FIFO is full</li> </ul> <p>Exists: IC_STAT_FOR_CLK_STRETCH == 1 Volatile: true</p>	0x0
SLV_HOLD_TX_FIFO_EMPTY	[9]	R	<p>This bit indicates the BUS Hold in Slave mode for the Read request when the Tx FIFO is empty. The Bus is in hold until the Tx FIFO has data to Transmit for the read request.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Slave holds the bus due to Tx FIFO is empty</li> <li>- 0x0 (INACTIVE): Slave is not holding the bus or Bus hold is not due to Tx FIFO is empty</li> </ul> <p>Exists: IC_STAT_FOR_CLK_STRETCH == 1 Volatile: true</p>	0x0

MST_HOLD_RX_FIFO_FULL	[8]	R	<p>This bit indicates the BUS Hold in Master mode due to Rx FIFO is Full and additional byte has been received (This kind of Bus hold is applicable if IC_RX_FULL_HLD_BUS_EN is set to 1).</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Master holds the bus due to Rx FIFO is full</li> <li>- 0x0 (INACTIVE): Master is not holding the bus or Bus hold is not due to Rx FIFO is full</li> </ul> <p>Exists: IC_STAT_FOR_CLK_STRETCH == 1 Volatile: true</p>	0x0
MST_HOLD_TX_FIFO_EMPTY	[7]	R	<p>If the IC_EMPTYFIFO_HOLD_MASTER_EN parameter is set to 1, the DW_apb_i2c master stalls the write transfer when Tx FIFO is empty, and the last byte does not have the Stop bit set. This bit indicates the BUS hold when the master holds the bus because of the Tx FIFO being empty, and the previous transferred command does not have the Stop bit set. (This kind of Bus hold is applicable if IC_EMPTYFIFO_HOLD_MASTER_EN is set to 1).</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Master holds the bus due to Tx FIFO is empty</li> <li>- 0x0 (INACTIVE): Master is not holding the bus or Bus hold is not due to Tx FIFO is empty</li> </ul> <p>Exists: IC_STAT_FOR_CLK_STRETCH == 1 Volatile: true</p>	0x0
SLV_ACTIVITY	[6]	R	<p>Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set.</p> <ul style="list-style-type: none"> <li>- 0: Slave FSM is in IDLE state so the Slave part of DW_apb_i2c is not Active</li> <li>- 1: Slave FSM is not in IDLE state so the Slave part of DW_apb_i2c is Active</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Slave not idle</li> <li>- 0x0 (IDLE): Slave is idle</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
MST_ACTIVITY	[5]	R	<p>Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set.</p> <ul style="list-style-type: none"> <li>- 0: Master FSM is in IDLE state so the Master part of DW_apb_i2c is not Active</li> <li>- 1: Master FSM is not in IDLE state so the Master part of DW_apb_i2c is Active</li> </ul> <p>Note: IC_STATUS[0]-that is, ACTIVITY bit-is the OR of SLV_ACTIVITY and MST_ACTIVITY bits.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Master not idle</li> <li>- 0x0 (IDLE): Master is idle</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
RFF	[4]	R	<p>Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared.</p> <ul style="list-style-type: none"> <li>- 0: Receive FIFO is not full</li> <li>- 1: Receive FIFO is full</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (FULL): Rx FIFO is full</li> <li>- 0x0 (NOT_FULL): Rx FIFO not full</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
RFNE	[3]	R	<p>Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty.</p> <ul style="list-style-type: none"> <li>- 0: Receive FIFO is empty</li> <li>- 1: Receive FIFO is not empty</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (NOT_EMPTY): Rx FIFO not empty</li> <li>- 0x0 (EMPTY): Rx FIFO is empty</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

TFE	[2]	R	<p>Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt.</p> <ul style="list-style-type: none"> <li>- 0: Transmit FIFO is not empty</li> <li>- 1: Transmit FIFO is empty</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (EMPTY): Tx FIFO is empty</li> <li>- 0x0 (NON_EMPTY): Tx FIFO not empty</li> </ul> <p>Exists: Always Volatile: true</p>	0x1
TFNF	[1]	R	<p>Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full.</p> <ul style="list-style-type: none"> <li>- 0: Transmit FIFO is full</li> <li>- 1: Transmit FIFO is not full</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (NOT_FULL): Tx FIFO not full</li> <li>- 0x0 (FULL): Tx FIFO is full</li> </ul> <p>Exists: Always Volatile: true</p>	0x1
ACTIVITY	[0]	R	<p>I2C Activity Status.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): I2C is active</li> <li>- 0x0 (INACTIVE): I2C is idle</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

### 15.5.34 I2C Transmit FIFO Level Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x74 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_TXFLR	[31:4]	R	<p>TXFLR Register field Reserved bits - Read Only</p> <p>Exists: Always Volatile: true Range Variable[y]: 4</p>	0x0000
TXFLR	[3:0]	R	<p>Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.</p> <p>Exists: Always Volatile: true Range Variable[x]: 3</p>	0x0

### 15.5.35 I2C Receive FIFO Level Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x78 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_RXFLR	[31:4]	R	RXFLR Reserved bits - Read Only Exists: Always Volatile: true Range Variable[y]: 4	0x0000
RXFLR	[3:0]	R	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. Exists: Always Volatile: true Range Variable[x]: 3	0x0

### 15.5.36 I2C SDA Hold Time Length Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x7C Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

RSVD_IC_SDA_HOLD	[31:24]	R	IC_SDA_HOLD Reserved bits - Read Only Exists: Always	0x0000
IC_SDA_RX_HOLD	[23:16]	R/W	Sets the required SDA hold time in units of ic_clk period, when DW_apb_i2c acts as a receiver.  Exists: Always	0x0
IC_SDA_TX_HOLD	[15:0]	R/W	Sets the required SDA hold time in units of ic_clk period, when DW_apb_i2c acts as a transmitter.  Exists: Always	0x1

### 15.5.37 I2C SDA Hold Time Length Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x80 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
TX_FLUSH_CNT	[31:23]	R	This field indicates the number of Tx FIFO Data Commands which are flushed due to TX_ABRT interrupt. It is cleared whenever I2C is disabled.  Role of DW_apb_i2c: Master-Transmitter or Slave- Transmitter Exists: Always Volatile: true	0x0
RSVD_IC_TX_ABRT_SOURCE	[22:21]	R	IC_TX_ABRT_SOURCE Reserved bits - Read Only Exists: Always Volatile: true	0x0000
ABRT_DEVICE_WRITE	[20]	R	This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the Tx-FIFO consists of write commands.  Role of DW_apb_i2c: Master Values: - 0x1 (ACTIVE): This abort is generated because of NOACK for Slave address - 0x0 (INACTIVE): This abort is not generated Exists: IC_DEVICE_ID == 1 Volatile: true	0x0

ABRT_DEVICE_SLVADDR_NOA CK	[19]	R	<p>This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the slave address sent was not acknowledged by any slave.</p> <p>Role of DW_apb_i2c: Master Values: - 0x1 (ACTIVE): This abort is generated because of NOACK for Slave address - 0x0 (INACTIVE): This abort is not generated Exists: IC_DEVICE_ID == 1 Volatile: true</p>	0x0
ABRT_DEVICE_NOACK	[18]	R	<p>This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the device id sent was not acknowledged by any slave.</p> <p>Role of DW_apb_i2c: Master Values: - 0x1 (ACTIVE): This abort is generated because of NOACK for DEVICE-ID - 0x0 (INACTIVE): This abort is not generated Exists: IC_DEVICE_ID == 1 Volatile: true</p>	0x0
ABRT_SDA_STUCK_AT_LOW	[17]	R	<p>This is a master-mode-only bit. Master detects the SDA Stuck at low for the IC_SDA_STUCK_AT_LOW_TIMEOUT value of ic_clks.</p> <p>Role of DW_apb_i2c: Master Values: - 0x1 (ACTIVE): This abort is generated because of Sda stuck at low for IC_SDA_STUCK_AT_LOW_TIMEOUT value of ic_clks - 0x0 (INACTIVE): This abort is not generated Exists: IC_BUS_CLEAR_FEATURE == 1 Volatile: true</p>	0x0
ABRT_USER_ABRT	[16]	R	<p>This is a master-mode-only bit. Master has detected the transfer abort (IC_ENABLE[1])</p> <p>Role of DW_apb_i2c: Master-Transmitter Values: - 0x1 (ABRT_USER_ABRT_GENERATED): Transfer abort detected by master - 0x0 (ABRT_USER_ABRT_VOID): Transfer abort detected by master- scenario not present Exists: Always Volatile: true</p>	0x0
ABRT_SLVRD_INTX	[15]	R	<p>1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of IC_DATA_CMD register.</p> <p>Role of DW_apb_i2c: Slave-Transmitter Values: - 0x1 (ABRT_SLVRD_INTX_GENERATED): Slave trying to transmit to remote master in read mode - 0x0 (ABRT_SLVRD_INTX_VOID): Slave trying to transmit to remote master in read mode- scenario not present Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
ABRT_SLV_ARBLOST	[14]	R	<p>This field indicates that a Slave has lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. Note: Even though the slave never 'owns' the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then DW_apb_i2c no longer own the bus.</p> <p>Role of DW_apb_i2c: Slave-Transmitter Values: - 0x1 (ABRT_SLV_ARBLOST_GENERATED): Slave lost arbitration to remote master - 0x0 (ABRT_SLV_ARBLOST_VOID): Slave lost arbitration to remote master- scenario not present Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0

ABRT_SLVFLUSH_TXFIFO	[13]	R	<p>This field specifies that the Slave has received a read command and some data exists in the TX FIFO, so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO.</p> <p>Role of DW_apb_i2c: Slave-Transmitter Values: - 0x1 (ABRT_SLVFLUSH_TXFIFO_GENERATED): Slave flushes existing data in TX-FIFO upon getting read command - 0x0 (ABRT_SLVFLUSH_TXFIFO_VOID): Slave flushes existing data in TX-FIFO upon getting read command- scenario not present Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
ARB_LOST	[12]	R	<p>This field specifies that the Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration.</p> <p>Role of DW_apb_i2c: Master-Transmitter or Slave- Transmitter Values: - 0x1 (ABRT_LOST_GENERATED): Master or Slave- Transmitter lost arbitration - 0x0 (ABRT_LOST_VOID): Master or Slave-Transmitter lost arbitration- scenario not present Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
ABRT_MASTER_DIS	[11]	R	<p>This field indicates that the User tries to initiate a Master operation with the Master mode disabled.</p> <p>Role of DW_apb_i2c: Master-Transmitter or Master- Receiver Values: - 0x1 (ABRT_MASTER_DIS_GENERATED): User initiating master operation when MASTER disabled - 0x0 (ABRT_MASTER_DIS_VOID): User initiating master operation when MASTER disabled- scenario not present Exists: Always Volatile: true</p>	0x0
ABRT_10B_RD_NORSTRT	[10]	R	<p>This field indicates that the restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the master sends a read command in 10-bit addressing mode.</p> <p>Role of DW_apb_i2c: Master-Receiver Values: - 0x1 (ABRT_10B_RD_GENERATED): Master trying to read in 10Bit addressing mode when RESTART disabled - 0x0 (ABRT_10B_RD_VOID): Master not trying to read in 10Bit addressing mode when RESTART disabled Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
ABRT_SBYTE_NORSTRT	[9]	R	<p>To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets reasserted. When this field is set to 1, the restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the user is trying to send a START Byte.</p> <p>Role of DW_apb_i2c: Master Values: - 0x1 (ABRT_SBYTE_NORSTRT_GENERATED): User trying to send START byte when RESTART disabled - 0x0 (ABRT_SBYTE_NORSTRT_VOID): User trying to send START byte when RESTART disabled- scenario not present Exists: Always Volatile: true</p>	0x0

ABRT_HS_NORSTRT	[8]	R	<p>This field indicates that the restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the user is trying to use the master to transfer data in High Speed mode.</p> <p>Role of DW_apb_i2c: Master-Transmitter or Master- Receiver Values:            - 0x1 (ABRT_HS_NORSTRT_GENERATED): User trying to switch Master to HS mode when RESTART disabled            - 0x0 (ABRT_HS_NORSTRT_VOID): User trying to switch Master to HS mode when RESTART disabled- scenario not present            Exists: IC_ULTRA_FAST_MODE==0            Volatile: true</p>	0x0
ABRT_SBYTE_ACKDET	[7]	R	<p>This field indicates that the Master has sent a START Byte and the START Byte was acknowledged (wrong behavior).</p> <p>Role of DW_apb_i2c: Master Values:            - 0x1 (ABRT_SBYTE_ACKDET_GENERATED): ACK detected for START byte            - 0x0 (ABRT_SBYTE_ACKDET_VOID): ACK detected for START byte- scenario not present            Exists: IC_ULTRA_FAST_MODE==0            Volatile: true</p>	0x0
ABRT_HS_ACKDET	[6]	R	<p>This field indicates that the Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior).</p> <p>Role of DW_apb_i2c: Master Values:            - 0x1 (ABRT_HS_ACK_GENERATED): HS Master code ACKed in HS Mode            - 0x0 (ABRT_HS_ACK_VOID): HS Master code ACKed in HS Mode- scenario not present            Exists: IC_ULTRA_FAST_MODE==0            Volatile: true</p>	0x0
ABRT_GCALL_READ	[5]	R	<p>This field indicates that DW_apb_i2c in the master mode has sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1).</p> <p>Role of DW_apb_i2c: Master-Transmitter Values:            - 0x1 (ABRT_GCALL_READ_GENERATED): GCALL is followed by read from bus            - 0x0 (ABRT_GCALL_READ_VOID): GCALL is followed by read from bus-scenario not present            Exists: IC_ULTRA_FAST_MODE==0            Volatile: true</p>	0x0
ABRT_GCALL_NOACK	[4]	R	<p>This field indicates that DW_apb_i2c in master mode has sent a General Call and no slave on the bus acknowledged the General Call.</p> <p>Role of DW_apb_i2c: Master-Transmitter Values:            - 0x1 (ABRT_GCALL_NOACK_GENERATED): GCALL not ACKed by any slave            - 0x0 (ABRT_GCALL_NOACK_VOID): GCALL not ACKed by any slave-scenario not present            Exists: IC_ULTRA_FAST_MODE==0            Volatile: true</p>	0x0
ABRT_TXDATA_NOACK	[3]	R	<p>This field indicates the master-mode only bit. When the master receives an acknowledgement for the address, but when it sends data byte(s) following the address, it did not receive an acknowledgement from the remote slave(s).</p> <p>Role of DW_apb_i2c: Master-Transmitter Values:            - 0x1 (ABRT_TXDATA_NOACK_GENERATED): Transmitted data not ACKed by addressed slave            - 0x0 (ABRT_TXDATA_NOACK_VOID): Transmitted data non-ACKed by addressed slave-scenario not present            Exists: IC_ULTRA_FAST_MODE==0            Volatile: true</p>	0x0

ABRT_10ADDR2_NOACK	[2]	R	<p>This field indicates that the Master is in 10-bit address mode and that the second address byte of the 10-bit address was not acknowledged by any slave.</p> <p>Role of DW_apb_i2c: Master-Transmitter or Master- Receiver Values: - 0x1 (ACTIVE): Byte 2 of 10Bit Address not ACKed by any slave - 0x0 (INACTIVE): This abort is not generated Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
ABRT_10ADDR1_NOACK	[1]	R	<p>This field indicates that the Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave.</p> <p>Role of DW_apb_i2c: Master-Transmitter or Master- Receiver Values: - 0x1 (ACTIVE): Byte 1 of 10Bit Address not ACKed by any slave - 0x0 (INACTIVE): This abort is not generated Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0
ABRT_7B_ADDR_NOACK	[0]	R	<p>This field indicates that the Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.</p> <p>Role of DW_apb_i2c: Master-Transmitter or Master- Receiver Values: - 0x1 (ACTIVE): This abort is generated because of NOACK for 7-bit address - 0x0 (INACTIVE): This abort is not generated Exists: IC_ULTRA_FAST_MODE==0 Volatile: true</p>	0x0

### 15.5.38 Generate Slave Data NACK Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x84 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_SLV_DATA_NACK_ON_LY	[31:1]	R	IC_SLV_DATA_NACK_ONLY Reserved bits - Read Only Exists: Always	0x0000

NACK	[0]	R/W	<p>Generate NACK. This NACK generation only occurs when DW_apb_i2c is a slave-receiver. If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence, the data transfer is aborted and the data received is not pushed to the receive buffer.</p> <p>When the register is set to a value of 0, it generates NACK/ACK, depending on normal criteria.</p> <ul style="list-style-type: none"> <li>- 1: generate NACK after data byte received</li> <li>- 0: generate NACK/ACK normally</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Slave receiver generates NACK upon data reception only</li> <li>- 0x0 (DISABLED): Slave receiver generates NACK normally</li> </ul> <p>Exists: Always</p>	0x0000
------	-----	-----	--	--------

### 15.5.39 DMA Control Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x88 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_DMA_CR_2_31	[31:2]	R	RSVD_IC_DMA_CR_2_31 Reserved bits - Read Only Exists: Always	0x0000
TDMAE	[1]	R/W	<p>Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Transmit FIFO DMA channel enabled</li> <li>- 0x0 (DISABLED): transmit FIFO DMA channel disabled</li> </ul> <p>Exists: Always</p>	0x0
RDMAE	[0]	R/W	<p>Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ENABLED): Receive FIFO DMA channel enabled</li> <li>- 0x0 (DISABLED): Receive FIFO DMA channel disabled</li> </ul> <p>Exists: Always</p>	0x0

### 15.5.40 DMA Transmit Data Level Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x8C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_DMA_TDLR	[31:3]	R	DMA_TDLR Reserved bits - Read Only Exists: Always Range Variable[y]: 3	0x0000
DMATDL	[2:0]	R/W	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.  Exists: Always Range Variable[x]: 2	0x0

### 15.5.41 DMA Transmit Data Level Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x90 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_DMA_RDLR	[31:3]	R	DMA_RDLR Reserved bits - Read Only Exists: Always Range Variable[y]:3	0x0000
DMARDL	[2:0]	R/W	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and RDMAE =1. For instance, when DMARDL is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.  Exists: Always Range Variable[x]: 2	0x0

### 15.5.42 I2C SDA Setup Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x94 Reset Value = 0x0000\_0064

Name	Bit	Type	Description	Reset Value
RSVD_IC_SDA_SETUP	[31:8]	R	IC_SDA_SETUP Reserved bits - Read Only Exists: Always	0x0000
SDA_SETUP	[7:0]	R/W	SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. IC_SDA_SETUP must be programmed with a minimum value of 2. but can be hardcoded by setting the IC_DEFAULT_SDA_SETUP configuration parameter. Exists: Always	0x64

### 15.5.43 I2C ACK General Call Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x98 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD_IC_ACK_GEN_1_31	[31:1]	R	RSVD_IC_ACK_GEN_1_31 Reserved bits - Read Only Exists: Always	0x0000
ACK_GEN_CALL	[0]	R/W	ACK General Call. When set to 1, DW_apb_i2c responds with a ACK (by asserting ic_data_oe) when it receives a General Call. Otherwise, DW_apb_i2c responds with a NACK (by negating ic_data_oe). , but can be hardcoded by setting the IC_DEFAULT_ACK_GENERAL_CALL configuration parameter. Values: - 0x1 (ENABLED): Generate ACK for a General Call - 0x0 (DISABLED): Generate NACK for General Call Exists: Always	0x1

### 15.5.44 I2C Enable Status Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0x9C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_ENABLE_STATUS	[31:3]	R	IC_ENABLE_STATUS Reserved bits - Read Only Exists: Always Volatile: true	0x0000
SLV_RX_DATA_LOST	[2]	R	Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to the setting bit 0 of IC_ENABLE from 1 to 0. When read as 1, DW_apb_i2c is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. Note: If the remote I2C master terminates the transfer with a STOP condition before the DW_apb_i2c has a chance to NACK a transfer, and IC_ENABLE[0] has been set to 0, then this bit is also set to 1. When read as 0, DW_apb_i2c is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer. Note: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.  Values: - 0x1 (ACTIVE): Slave RX Data is lost - 0x0 (INACTIVE): Slave RX Data is not lost Exists: Always Volatile: true	0x0
SLV_DISABLED_WHILE_BUSY	[1]	R	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting bit 0 of the IC_ENABLE register from 1 to 0. This bit is set when the CPU writes a 0 to the IC_ENABLE register while: (a) DW_apb_i2c is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master. When read as 1, DW_apb_i2c is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in DW_apb_i2c (IC_SAR register) OR if the transfer is completed before IC_ENABLE is set to 0 but has not taken effect. Note: If the remote I2C master terminates the transfer with a STOP condition before the DW_apb_i2c has a chance to NACK a transfer, and IC_ENABLE[0] has been set to 0, then this bit will also be set to 1. When read as 0, DW_apb_i2c is deemed to have been disabled when there is master activity, or when the I2C bus is idle. Note: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.  Values: - 0x1 (ACTIVE): Slave is disabled when it is active - 0x0 (INACTIVE): Slave is disabled when it is idle Exists: Always Volatile: true	0x0
IC_EN	[0]	R	ic_en Status. This bit always reflects the value driven on the output port ic_en. - When read as 1, DW_apb_i2c is deemed to be in an enabled state. - When read as 0, DW_apb_i2c is deemed completely inactive. Note: The CPU can safely read this bit anytime. When this bit is read as 0, the CPU can safely read SLV_RX_DATA_LOST (bit 2) and SLV_DISABLED_WHILE_BUSY (bit 1).  Values: - 0x1 (ENABLED): I2C enabled - 0x0 (DISABLED): I2C disabled Exists: Always Volatile: true	0x0

### 15.5.45 I2C SS, FS or FM+ spike suppression limit

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xA0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

### 15.5.46 I2C Ultra-Fast mode spike suppression limit

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xA0 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD_IC_UFM_SP-KLEN	[31:8]	R	IC_UFM_SPKLEN Reserved bits - Read Only Exists: Always	0x0000

IC_UFM_SPKLEN	[7:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic.</p> <p>This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 1; hardware prevents values less than this being written, and if attempted results in 1 being set.</p> <p>Exists: Always</p>	0x1
---------------	-------	-----	---	-----

### 15.5.47 I2C HS spike suppression limit register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xA4 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD_IC_HS_SPKLEN	[31:8]	R	IC_HS_SPKLEN Reserved bits - Read Only Exists: Always	0x0000
IC_HS_SPKLEN	[7:0]	R/W	<p>This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic; for more information, refer to "Spike Suppression".</p> <p>This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 1; hardware prevents values less than this being written, and if attempted results in 1 being set.</p> <p>Exists: Always</p>	0x1

### 15.5.48 Clear RESTART\_DET Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xA8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_CLR_RESTART_DET	[31:1]	R	IC_CLR_RESTART_DET Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_RESTART_DET	[0]	R	Read this register to clear the RESTART_DET interrupt (bit 12) of IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.49 I2C SCL Stuck at Low Timeout register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xAC Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

IC_SCL_STUCK_LOW_TIMEOUT	[31:0]	R/W	DW_apb_i2c generate the interrupt to indicate SCL stuck at low (SCL_STUCK_AT_LOW) if it detects the SCL stuck at low for the IC_SCL_STUCK_LOW_TIMEOUT in units of ic_clk period. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE[0] register being set to 0. Writes at other times have no effect.  Exists: Always	0xffffffff
--------------------------	--------	-----	--	------------

### 15.5.50 I2C SDA Stuck at Low Timeout register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xB0 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
IC_SDA_STUCK_LOW_TIMEOUT	[31:0]	R/W	DW_apb_i2c initiates the recovery of SDA line through enabling the SDA_STUCK_RECOVERY_EN (IC_ENABLE[3]) register bit, if it detects the SDA stuck at low for the IC_SDA_STUCK_LOW_TIMEOUT in units of ic_clk period.  Exists: Always	0xffffffff

### 15.5.51 Clear SCL Stuck at Low Detect interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)

- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xB4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_CLR_SCL_STUCK_DET	[31:1]	R	CLR_SCL_STUCK_DET Reserved bits - Read Only Exists: Always Volatile: true	0x0000
CLR_SCL_STUCK_DET	[0]	R	Read this register to clear the SCL_STUCK_AT_LOW interrupt (bit 15) of the IC_RAW_INTR_STAT register. Exists: Always Volatile: true	0x0

### 15.5.52 I2C Device-ID Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xB8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_DEVICE_ID	[31:24]	R	IC_DEVICE_ID Reserved bits - Read Only Exists: Always	0x0000
DEVICE-ID	[23:0]	R	Contains the Device-ID of the component assigned through the configuration parameter 'IC_DEVICE_ID_VALUE' Exists: Always	0x0

### 15.5.53 SMBus Slave Clock Extend Timeout register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xBC Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
SMBUS_CLK_LOW_SEXT_TIME_OUT	[31:0]	R/W	This field is used to detect the Slave Clock Extend timeout (tLOW:SEXT) in master mode extended by the slave device in one message from the initial START to the STOP. The values in this register are in units of ic_clk period.  Exists: Always	0xffffffff

### 15.5.54 SMBus Master Clock Extend Timeout register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xC0 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

SMBUS_CLK_LOW_MEXT_TIME_OUT	[31:0]	R/W	This field is used to detect the Master extend SMBus clock (SCLK) timeout defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP in Master mode. The values in this register are in units of ic_clk period.  Exists: Always	0xffffffff
-----------------------------	--------	-----	--	------------

### 15.5.55 SMBus Master THigh MAX Bus-idle count Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xC4 Reset Value = 0x0000\_FFFF

Name	Bit	Type	Description	Reset Value
RSVD_SMBUS_THIGH_MAX_BU_S_IDLE_CNT	[31:16]	R	SMBUS_THIGH_MAX_BUS_IDLE_CNT Reserved bits - Read Only  Exists: Always	0x0000
SMBUS_THIGH_MAX_BUS_IDL_E_CNT	[15:0]	R/W	This field is used to set the required Bus-Idle time period used when a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines.  In this case, the master must wait long enough to ensure that a transfer is not currently in progress The values in this register are in units of ic_clk period.  Exists: Always	0xffff

### 15.5.56 SMBus Interrupt Status Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)

- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xC8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_SMBUS_INTR_STAT	[31:11]	R	IC_SMBUS_INTR_STAT Reserved bits - Read Only Exists: Always Volatile: true	0x0000
R_SMBUS_ALERT_DET	[10]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_SMBUS_ALERT_DET bit.  Values: - 0x1 (ACTIVE): SMBUS_ALERT_DET interrupt is active - 0x0 (INACTIVE): SMBUS_ALERT_DET interrupt is inactive Exists: IC_SMBUS_SUSPEND_ALERT==1 Volatile: true	0x0
R_SMBUS_SUSPEND_DET	[9]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_SMBUS_SUSPEND_DET bit.  Values: - 0x1 (ACTIVE): SMBUS_SUSPEND_DET interrupt is active - 0x0 (INACTIVE): SMBUS_SUSPEND_DET interrupt is inactive Exists: IC_SMBUS_SUSPEND_ALERT==1 Volatile: true	0x0
R_SLV_RX_PEC_NACK	[8]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_SLV_RX_PEC_NACK bit.  Values: - 0x1 (ACTIVE): SLV_RX_PEC_NACK interrupt is active - 0x0 (INACTIVE): SLV_RX_PEC_NACK interrupt is inactive Exists: IC_SMBUS_ARP==1 Volatile: true	0x0
R_ARP_ASSGN_ADDR_CMD_DET	[7]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_ARP_ASSGN_ADDR_CMD_DET bit. Values: - 0x1 (ACTIVE): ARP_ASSGN_ADDR_CMD_DET interrupt is active - 0x0 (INACTIVE): ARP_ASSGN_ADDR_CMD_DET interrupt is inactive Exists: IC_SMBUS_ARP==1 Volatile: true	0x0
R_ARP_GET_UDID_CMD_DET	[6]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_ARP_GET_UDID_CMD_DET bit.  Values: - 0x1 (ACTIVE): ARP_GET_UDID_CMD_DET interrupt is active - 0x0 (INACTIVE): ARP_GET_UDID_CMD_DET interrupt is inactive Exists: IC_SMBUS_ARP==1 Volatile: true	0x0
R_ARP_RST_CMD_DET	[5]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_ARP_RST_CMD_DET bit.  Values: - 0x1 (ACTIVE): ARP_RST_CMD_DET interrupt is active - 0x0 (INACTIVE): ARP_RST_CMD_DET interrupt is inactive Exists: IC_SMBUS_ARP==1 Volatile: true	0x0
R_ARP_PREPARE_CMD_DET	[4]	R	See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_ARP_PREPARE_CMD_DET bit.  Values: - 0x1 (ACTIVE): ARP_PREPARE_CMD_DET interrupt is active - 0x0 (INACTIVE): ARP_PREPARE_CMD_DET interrupt is inactive Exists: IC_SMBUS_ARP==1 Volatile: true	0x0

R_HOST_NOTIFY_MST_DET	[3]	R	<p>See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_HOST_NOTIFY_MST_DET bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): HOST_NOTIFY_MST_DET interrupt is active</li> <li>- 0x0 (INACTIVE): HOST_NOTIFY_MST_DET interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS==1 Volatile: true</p>	0x0
R_QUICK_CMD_DET	[2]	R	<p>See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_QUICK_CMD_DET bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): QUICK_CMD_DET interrupt is active</li> <li>- 0x0 (INACTIVE): QUICK_CMD_DET interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
R_MST_CLOCK_EXTND_TIMEOUT_UT	[1]	R	<p>See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_MST_CLOCK_EXTND_TIMEOUT bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): MST_CLOCK_EXTND_TIMEOUT interrupt is active</li> <li>- 0x0 (INACTIVE): MST_CLOCK_EXTND_TIMEOUT interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
R_SLV_CLOCK_EXTND_TIMEOUT_UT	[0]	R	<p>See IC_SMBUS_INTR_RAW_STATUS for a detailed description of R_SLV_CLOCK_EXTND_TIMEOUT bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SLV_CLOCK_EXTND_TIMEOUT interrupt is active</li> <li>- 0x0 (INACTIVE): SLV_CLOCK_EXTND_TIMEOUT interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

### 15.5.57 SMBus Interrupt Mask Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xCC Reset Value = 0x0000\_07FF

Name	Bit	Type	Description	Reset Value
RSVD_IC_SMBUS_INTR_MASK	[31:11]	R	IC_SMBUS_INTR_MASK Reserved bits - Read Only Exists: Always	0x0000

M_SMBUS_ALERT_DET	[10]	R/W	<p>This bit masks the R_SMBUS_ALERT_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): SMBUS_ALERT_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): SMBUS_ALERT_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_SUSPEND_ALERT==1</p>	0x1
M_SMBUS_SUSPEND_DET	[9]	R/W	<p>This bit masks the R_SMBUS_SUSPEND_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): SMBUS_SUSPEND_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): SMBUS_SUSPEND_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_SUSPEND_ALERT==1</p>	0x1
M_SLV_RX_PEC_NACK	[8]	R/W	<p>This bit masks the R_SLV_RX_PEC_NACK interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): SLV_RX_PEC_NACK interrupt is unmasked</li> <li>- 0x0 (ENABLED): SLV_RX_PEC_NACK interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_ARP==1</p>	0x1
M_ARP_ASSGN_ADDR_CMD_DET	[7]	R/W	<p>This bit masks the R_ARP_ASSGN_ADDR_CMD_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): ARP_ASSGN_ADDR_CMD_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): ARP_ASSGN_ADDR_CMD_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_ARP==1</p>	0x1
M_ARP_GET_UDID_CMD_DET	[6]	R/W	<p>This bit masks the R_ARP_GET_UDID_CMD_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): ARP_GET_UDID_CMD_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): ARP_GET_UDID_CMD_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_ARP==1</p>	0x1
M_ARP_RST_CMD_DET	[5]	R/W	<p>This bit masks the R_ARP_RST_CMD_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): ARP_RST_CMD_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): ARP_RST_CMD_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_ARP==1</p>	0x1
M_ARP_PREPARE_CMD_DET	[4]	R/W	<p>This bit masks the R_ARP_PREPARE_CMD_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): ARP_PREPARE_CMD_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): ARP_PREPARE_CMD_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS_ARP==1</p>	0x1
M_HOST_NOTIFY_MST_DET	[3]	R/W	<p>This bit masks the R_HOST_NOTIFY_MST_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): HOST_NOTIFY_MST_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): HOST_NOTIFY_MST_DET interrupt is masked</li> </ul> <p>Exists: IC_SMBUS==1</p>	0x1
M_QUICK_CMD_DET	[2]	R/W	<p>This bit masks the R_QUICK_CMD_DET interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): QUICK_CMD_DET interrupt is unmasked</li> <li>- 0x0 (ENABLED): QUICK_CMD_DET interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1

M_MST_CLOCK_EXTND_TIMEOUT	[1]	R/W	<p>This bit masks the R_MST_CLOCK_EXTND_TIMEOUT interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): MST_CLOCK_EXTND_TIMEOUT interrupt is unmasked</li> <li>- 0x0 (ENABLED): MST_CLOCK_EXTND_TIMEOUT interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1
M_SLV_CLOCK_EXTND_TIMEOUT	[0]	R/W	<p>This bit masks the R_SLV_CLOCK_EXTND_TIMEOUT interrupt in IC_SMBUS_INTR_STAT register.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): SLV_CLOCK_EXTND_TIMEOUT interrupt is unmasked</li> <li>- 0x0 (ENABLED): SLV_CLOCK_EXTND_TIMEOUT interrupt is masked</li> </ul> <p>Exists: Always</p>	0x1

### 15.5.58 SMBus Raw Interrupt Status Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xD0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_SMBUS_RAW_INTR_STAT	[31:11]	R	IC_SMBUS_RAW_INTR_STAT Reserved bits - Read Only Exists: Always Volatile: true	0x0000
SMBUS_ALERT_DET	[10]	R	Indicates whether a SMBALERT (ic_smbalert_in_n) signal is driven low by the slave.  Values: - 0x1 (ACTIVE): SMBUS Alert interrupt is active - 0x0 (INACTIVE): SMBUS Alert interrupt is inactive Exists: IC_SMBUS_SUSPEND_ALERT==1 Volatile: true	0x0
SMBUS_SUSPEND_DET	[9]	R	Indicates whether a SMBSSUS (ic_smbsus_in_n) signal is driven low by the Host.  Values: - 0x1 (ACTIVE): SMBUS System Suspended interrupt is active - 0x0 (INACTIVE): SMBUS System Suspended interrupt is inactive Exists: IC_SMBUS_SUSPEND_ALERT==1 Volatile: true	0x0

SLV_RX_PEC_NACK	[8]	R	<p>Indicates whether a NACK has been sent due to PEC mismatch while working as ARP slave.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): SLV_RX_PEC_NACK interrupt is active</li> <li>- 0x0 (INACTIVE): SLV_RX_PEC_NACK interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
ARP_ASSGN_ADDR_CMD_DET	[7]	R	<p>Indicates whether an Assign Address ARP command has been received.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): ARP_ASSGN_ADDR_CMD_DET interrupt is active</li> <li>- 0x0 (INACTIVE): ARP_ASSGN_ADDR_CMD_DET interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
ARP_GET_UDID_CMD_DET	[6]	R	<p>Indicates whether a Get UDID ARP command has been received.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): ARP_GET_UDID_CMD_DET interrupt is active</li> <li>- 0x0 (INACTIVE): ARP_GET_UDID_CMD_DET interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
ARP_RST_CMD_DET	[5]	R	<p>Indicates whether a General or Directed Reset ARP command has been received.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): ARP_RST_CMD_DET interrupt is active</li> <li>- 0x0 (INACTIVE): ARP_RST_CMD_DET interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
ARP_PREPARE_CMD_DET	[4]	R	<p>Indicates whether a prepare to ARP command has been received.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): ARP_PREPARE_CMD_DET interrupt is active</li> <li>- 0x0 (INACTIVE): ARP_PREPARE_CMD_DET interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS_ARP==1 Volatile: true</p>	0x0
HOST_NTFY_MST_DET	[3]	R	<p>Indicates whether a Notify ARP Master ARP command has been received.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): HOST_NTFY_MST_DET interrupt is active</li> <li>- 0x0 (INACTIVE): HOST_NTFY_MST_DET interrupt is inactive</li> </ul> <p>Exists: IC_SMBUS==1 Volatile: true</p>	0x0
QUICK_CMD_DET	[2]	R	<p>Indicates whether a Quick command has been received on the SMBus interface regardless of whether DW_apb_i2c is operating in slave or master mode. Enabled only when IC_SMBUS=1 is set to 1.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Quick Command interrupt is active</li> <li>- 0x0 (INACTIVE): Quick Command interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

MST_CLOCK_EXTND_TIMEOUT	[1]	R	<p>Indicates whether the Master device transaction (START-to-ACK, ACK-to-ACK, or ACK-to-STOP) from START to STOP exceeds IC_SMBUS_CLOCK_LOW_MEXT time with in each byte of message.</p> <p>This bit is enabled only when:</p> <ul style="list-style-type: none"> <li>- IC_SMBUS=1</li> <li>- IC_CON[0]=1</li> <li>- IC_EMPTYFIFO_HOLD_MASTER_EN=1 or</li> <li>- IC_RX_FULL_HLD_BUS_EN=1</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Master Clock Extend Timeout interrupt is active</li> <li>- 0x0 (INACTIVE): Master Clock Extend Timeout interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0
SLV_CLOCK_EXTND_TIMEOUT	[0]	R	<p>Indicates whether the transaction from Slave (i.e from START to STOP) exceeds IC_SMBUS_CLK_LOW_SEXT time.</p> <p>This bit is enabled only when:</p> <ul style="list-style-type: none"> <li>- IC_SMBUS=1</li> <li>- IC_CON[0]=1</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (ACTIVE): Slave Clock Extend Timeout interrupt is active</li> <li>- 0x0 (INACTIVE): Slave Clock Extend Timeout interrupt is inactive</li> </ul> <p>Exists: Always Volatile: true</p>	0x0

### 15.5.59 Clear SMBus Interrupt Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xD4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
W	[31:11]	R/W	IC_CLR_SMBUS_INTR Reserved bits - Read Only Exists: Always	0x0000
W	[10]	R/W	Write this register bit to clear the SMBUS_ALERT_DET interrupt (bit 10) of the IC_SMBUS_RAW_INTR_STAT register.  Exists: IC_SMBUS_SUSPEND_ALERT==1	0x0
W	[9]	R/W	Write this register bit to clear the SMBUS_SUSPEND_DET interrupt (bit 9) of the IC_SMBUS_RAW_INTR_STAT register.  Exists: IC_SMBUS_SUSPEND_ALERT==1	0x0

W	[8]	R/W	Write this register bit to clear the SLV_RX_PEC_NACK interrupt (bit 8) of the IC_SMBUS_RAW_INTR_STAT register. Exists: IC_SMBUS_ARP==1	0x0
W	[7]	R/W	Write this register bit to clear the ARP_ASSGN_ADDR_CMD_DET interrupt (bit 7) of the IC_SMBUS_RAW_INTR_STAT register. Exists: IC_SMBUS_ARP==1	0x0
W	[6]	R/W	Write this register bit to clear the ARP_GET_UIDID_CMD_DET interrupt (bit 6) of the IC_SMBUS_RAW_INTR_STAT register. Exists: IC_SMBUS_ARP==1	0x0
W	[5]	R/W	Write this register bit to clear the ARP_RST_CMD_DET interrupt (bit 5) of the IC_SMBUS_RAW_INTR_STAT register. Exists: IC_SMBUS_ARP==1	0x0
W	[4]	R/W	Write this register bit to clear the ARP_PREPARE_CMD_DET interrupt (bit 4) of the IC_SMBUS_RAW_INTR_STAT register. Exists: IC_SMBUS_ARP==1	0x0
W	[3]	R/W	Write this register bit to clear the HOST_NOTIFY_MST_DET interrupt (bit 3) of the IC_SMBUS_RAW_INTR_STAT register. Exists: IC_SMBUS==1	0x0
W	[2]	R/W	Write this register bit to clear the QUICK_CMD_DET interrupt (bit 2) of the IC_SMBUS_RAW_INTR_STAT register. Exists: Always	0x0
W	[1]	R/W	Write this register bit to clear the MST_CLOCK_EXTND_TIMEOUT interrupt (bit 1) of the IC_SMBUS_RAW_INTR_STAT register. Exists: Always	0x0
W	[0]	R/W	Write this register bit to clear the SLV_CLOCK_EXTND_TIMEOUT interrupt (bit 0) of the IC_SMBUS_RAW_INTR_STAT register. Exists: Always	0x0

### 15.5.60 I2C Optional Slave Address Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xD8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_OPTIONAL_SAR	[31:7]	R	IC_OPTIONAL_SAR Reserved bits - Read Only Exists: Always	0x0000
OPTIONAL_SAR	[6:0]	R/W	Optional Slave address for DW_apb_i2c when operating as a slave in SMBus Mode. Exists: Always	0x0

### 15.5.61 SMBUS ARP UDID LSB Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xDC Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
SMBUS_UDID_LSB	[31:0]	R/W	This field is used to store the LSB 32 bit value of slave unique device identifier used in Address Resolution Protocol. Exists: Always	0xffffffff

### 15.5.62 Component Parameter Register 1

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)

- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xF4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD_IC_COMP_PARAM_1	[31:24]	R	IC_COMP_PARAM_1 Reserved bits - Read Only Exists: Always	0x0000
TX_BUFFER_DEPTH	[23:16]	R	The value of this register is derived from the IC_TX_BUFFER_DEPTH coreConsultant parameter. - 0x00 = Reserved - 0x01 = 2 - 0x02 = 3 - ... - 0xFF = 256 Exists: Always	0x0000
RX_BUFFER_DEPTH	[15:8]	R	The value of this register is derived from the IC_RX_BUFFER_DEPTH coreConsultant parameter. - 0x00: Reserved - 0x01: 2 - 0x02: 3 - ... - 0xFF: 256 Exists: Always	0x0000
ADD_ENCODED_PARAMS	[7]	R	The value of this register is derived from the IC_ADD_ENCODED_PARAMS coreConsultant parameter. Reading 1 in this bit means that the capability of reading these encoded parameters via software has been included. Otherwise, the entire register is 0 regardless of the setting of any other parameters that are encoded in the bits. Values: - 0x1 (ENABLED): Enables capability of reading encoded parameters - 0x0 (DISABLED): Disables capability of reading encoded parameters Exists: Always	0x0000
HAS_DMA	[6]	R	The value of this register is derived from the IC_HAS_DMA core-Consultant parameter. Values: - 0x1 (ENABLED): DMA handshaking signals are enabled - 0x0 (DISABLED): DMA handshaking signals are disabled Exists: Always	0x0000
INTR_IO	[5]	R	The value of this register is derived from the IC_INTR_IO core-Consultant parameter. Values: - 0x1 (COMBINED): COMBINED Interrupt outputs - 0x0 (INDIVIDUAL): INDIVIDUAL Interrupt outputs Exists: Always	0x0000
HC_COUNT_VALUES	[4]	R	The value of this register is derived from the IC_HC_COUNT VALUES coreConsultant parameter. Values: - 0x1 (ENABLED): Hard code the count values for each mode. - 0x0 (DISABLED): Programmable count values for each mode. Exists: Always	0x0000
MAX_SPEED_MODE	[3:2]	R	The value of this register is derived from the IC_MAX_SPEED_MODE coreConsultant parameter. - 0x0: Reserved - 0x1: Standard - 0x2: Fast - 0x3: High Values: - 0x1 (STANDARD): IC MAX SPEED is STANDARD MODE - 0x2 (FAST): IC MAX SPEED is FAST MODE - 0x3 (HIGH): IC MAX SPEED is HIGH MODE Exists: IC_ULTRA_FAST_MODE==0	0x0000
APB_DATA_WIDTH	[1:0]	R	The value of this register is derived from the APB_DATA_WIDTH coreConsultant parameter. Values: - 0x0 (APB_08BITS): APB data bus width is 08 bits - 0x1 (APB_16BITS): APB data bus width is 16 bits - 0x2 (APB_32BITS): APB data bus width is 32 bits - 0x3 (RESERVED): Reserved bits Exists: Always	0x0000

### 15.5.63 I2C Component Version Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xF8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
IC_COMP_VERSION	[31:0]	R	Specific values for this register are described in the Releases Table in the DW_apb_i2c Release Notes Exists: Always	0x0000

### 15.5.64 I2C Component Type Register

- Base Address: 20600000(I2C0)
- Base Address: 20610000(I2C1)
- Base Address: 20620000(I2C2)
- Base Address: 20630000(I2C3)
- Base Address: 20640000(I2C4)
- Base Address: 20650000(I2C5)
- Base Address: 20660000(I2C6)
- Base Address: 20670000(I2C7)
- Base Address: 20680000(I2C8)
- Base Address: 20690000(I2C9)
- Base Address: 206A0000(I2C10)
- Base Address: 206B0000(I2C11)
- Address = Base Address + 0xFC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
IC_COMP_TYPE	[31:0]	R	Designware Component Type number = 0x44_57_01_40. This assigned unique hex value is constant and is derived from the two ASCII letters 'DW' followed by a 16-bit unsigned number. Exists: Always	0x0000

## 15.6 Programming the DW\_apb\_i2c

The DW\_apb\_i2c can be programmed via software registers or the DW\_apb\_i2c low-level software driver.

### 15.6.1 Software Registers

For information about programming the software registers in terms of DW\_apb\_i2c operation, refer to "Slave Mode Operation" and "Master Mode Operation". The software registers are described in more detail in "Register Descriptions".

### 15.6.2 Software Drivers

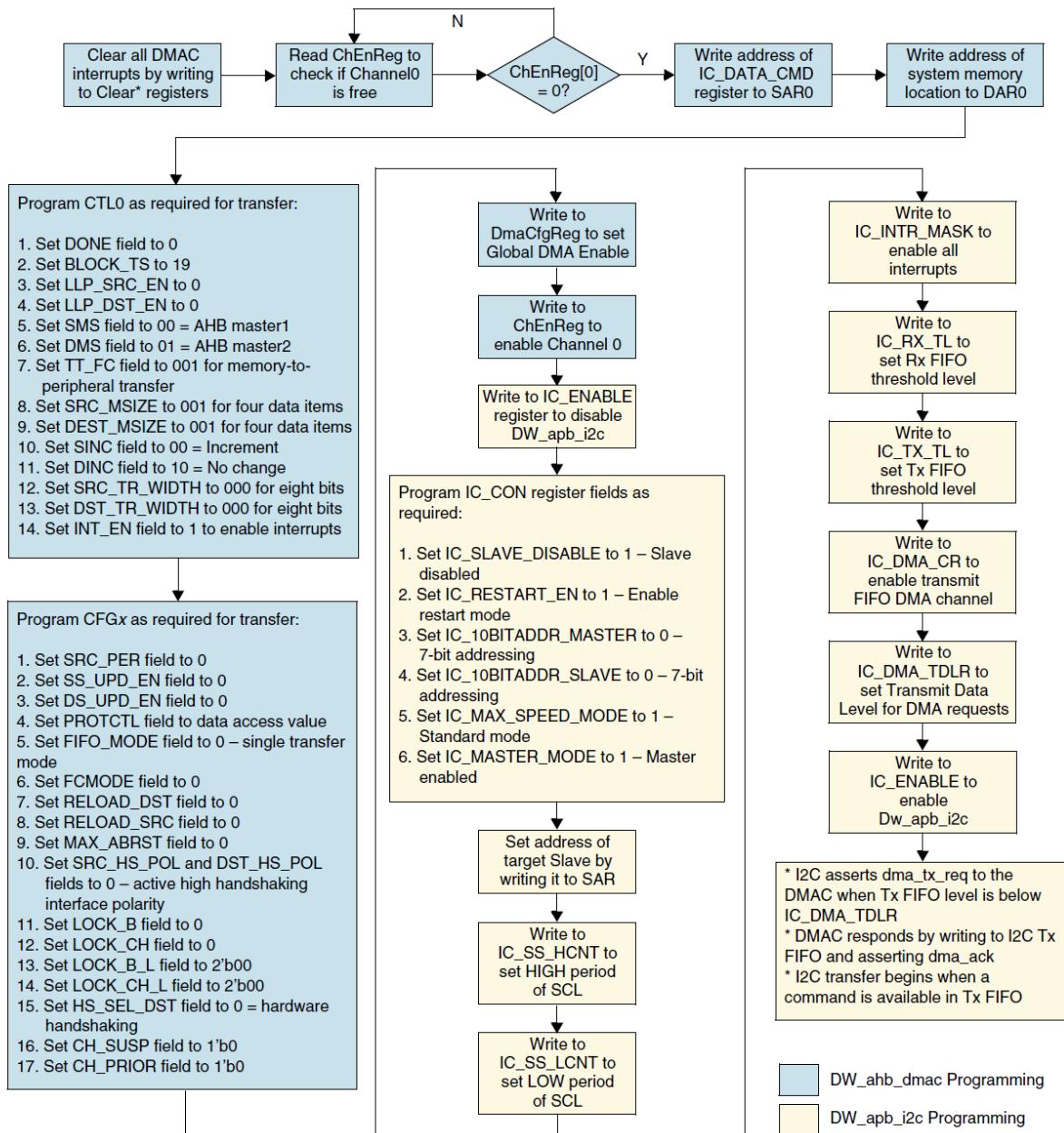
The family of DesignWare Synthesizable Components includes a Driver Kit for the DW\_apb\_i2c component. This low-level Driver Kit allows you to easily program a DW\_apb\_i2c component and integrate your code into a larger software system. The Driver Kit provides the following benefits to IP designers:

- Proven method of access to DW\_apb\_i2c minimizing usage errors
- Rapid software development with minimum overhead
- Detailed knowledge of DW\_apb\_i2c register bit fields not required
- Easy integration of DW\_apb\_i2c into existing software system
- Programming at register level eliminated

You must purchase a source code license (DWC-APB-Advanced-Source) to use the DW\_apb\_i2c Driver Kit. However, you can access some Driver Kit files and documentation in \$DESIGNWARE\_HOME/drivers/DW\_apb\_i2c/latest. For more information about the Driver Kit, refer to the DW\_apb\_i2c Driver Kit User Guide. For more information about purchasing the source code license and obtaining a download of the Driver Kit, contact Synopsys at designware@synopsys.com for details.

### 15.6.3 Programming Example

The flow diagram in Figure 15.55 shows an overview of programming the DW\_apb\_i2c.



**Figure 15.55:** Flowchart for DW\_ahb\_dmac and DW\_apb\_i2c Programming Example

#### Note

When there is at least one entry in the DW\_apb\_i2c Rx FIFO, the DW\_apb\_i2c asserts dma\_single to the DMAC. When the number of entries in the DW\_apb\_i2c Rx FIFO reaches reaches IC\_DMA\_RDLR, the DW\_apb\_i2c asserts dma\_rx\_req to the DMAC. In this example, in order to read nineteen data items from the DW\_apb\_i2c Rx

FIFO, the DMAC samples dma\_req for three BURST transfers of four beats of size 1 byte each, and it samples dma\_single for three SINGLE transfers of size 1 byte each.

The following outlines details regarding reads and writes to/from DW\_apb\_i2c masters/slaves and VIP master/slaves:

- For DW\_apb\_i2c master writes to a slave VIP model, bear in mind when using the DMA that you are writing characters from the byte stream. However, for a write, the DW\_apb\_i2c needs a halfword. To use the DMA, you should write software similar to the following: short int temp\_array[]; char \* ptr=(char \*) temp\_array; foreach byte in bytes store byte ptr++; store '0x01' write command ptr++
  - (a) Program the DMA to read a stream of halfwords from memory and write them to the DW\_apb\_i2c using the hardware interface.
  - (b) Program the DW\_apb\_i2c to do a write using the transmit DMA.
- For DW\_apb\_i2c master reads from a slave VIP model:
  - (a) Create a read command halfword.
  - (b) Program DMA channel 0 to do a fixed read of the read command halfword— that is, no address increment—to the DW\_apb\_i2c transmit buffer.
  - (c) Program DMA channel 1 to read the data from the read buffer and store it in memory.
  - (d) Program the DW\_apb\_i2c to do a master read by setting both DMA channels.
- For DW\_apb\_i2c slave writes from a master VIP model:
  - (a) Program the DW\_apb\_i2c to be a slave with the RX buffer DMA enabled.
  - (b) Program the DMA to read the buffer and store the bytes in memory.
- For DW\_apb\_i2c slave reads from a master VIP model:
  - (a) Enable IC\_INTR\_MASK.RD\_REQ; otherwise the DW\_apb\_i2c will not acknowledge the read.
  - (b) When you get the RD\_REQ interrupt, program the DMA to write the TX buffer with the read data.
  - (c) Program the DW\_apb\_i2c to enable the TX DMA.

The flow diagram in Figure 15.56 shows a programming example for the DW\_apb\_i2c Master.

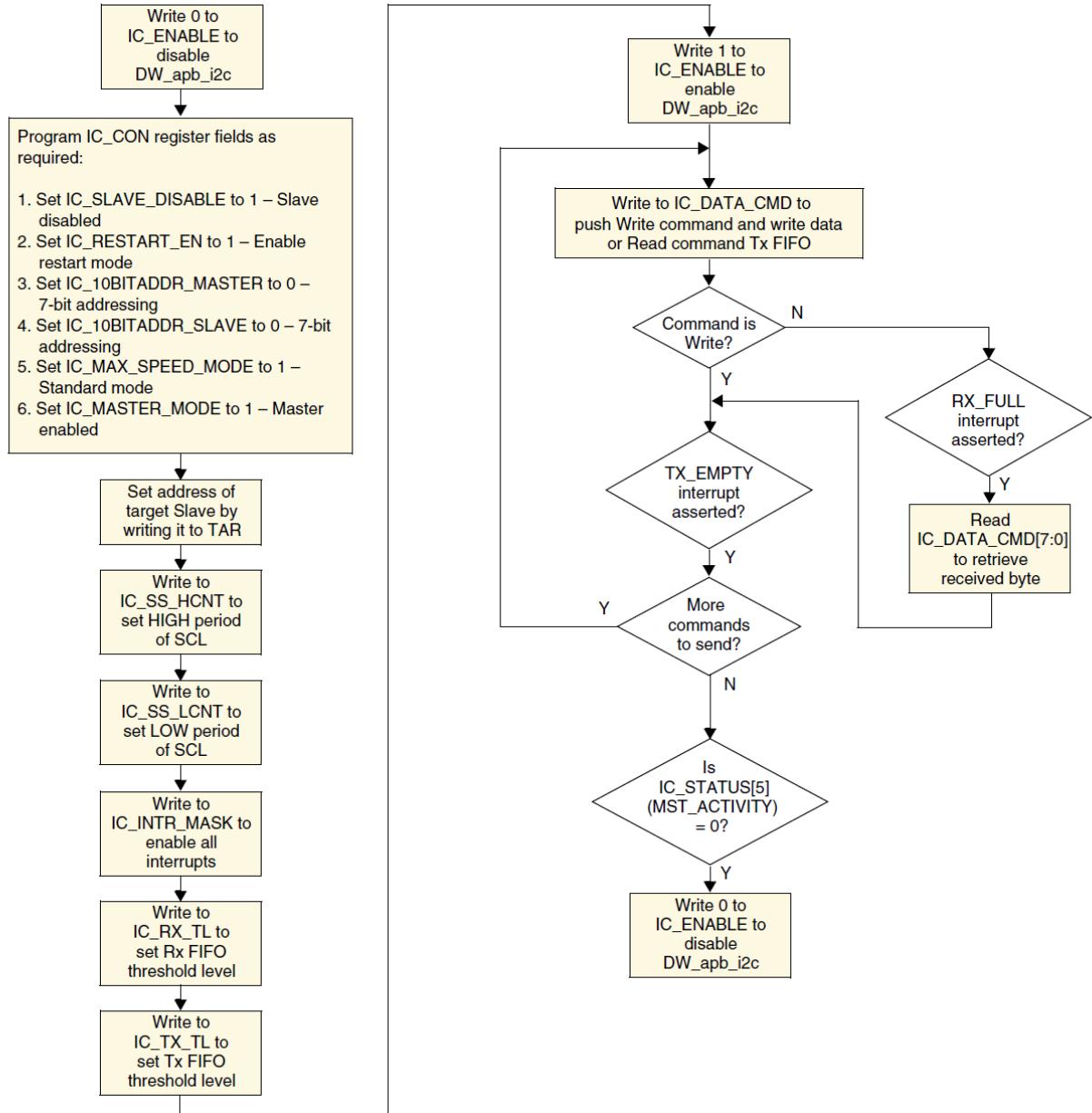
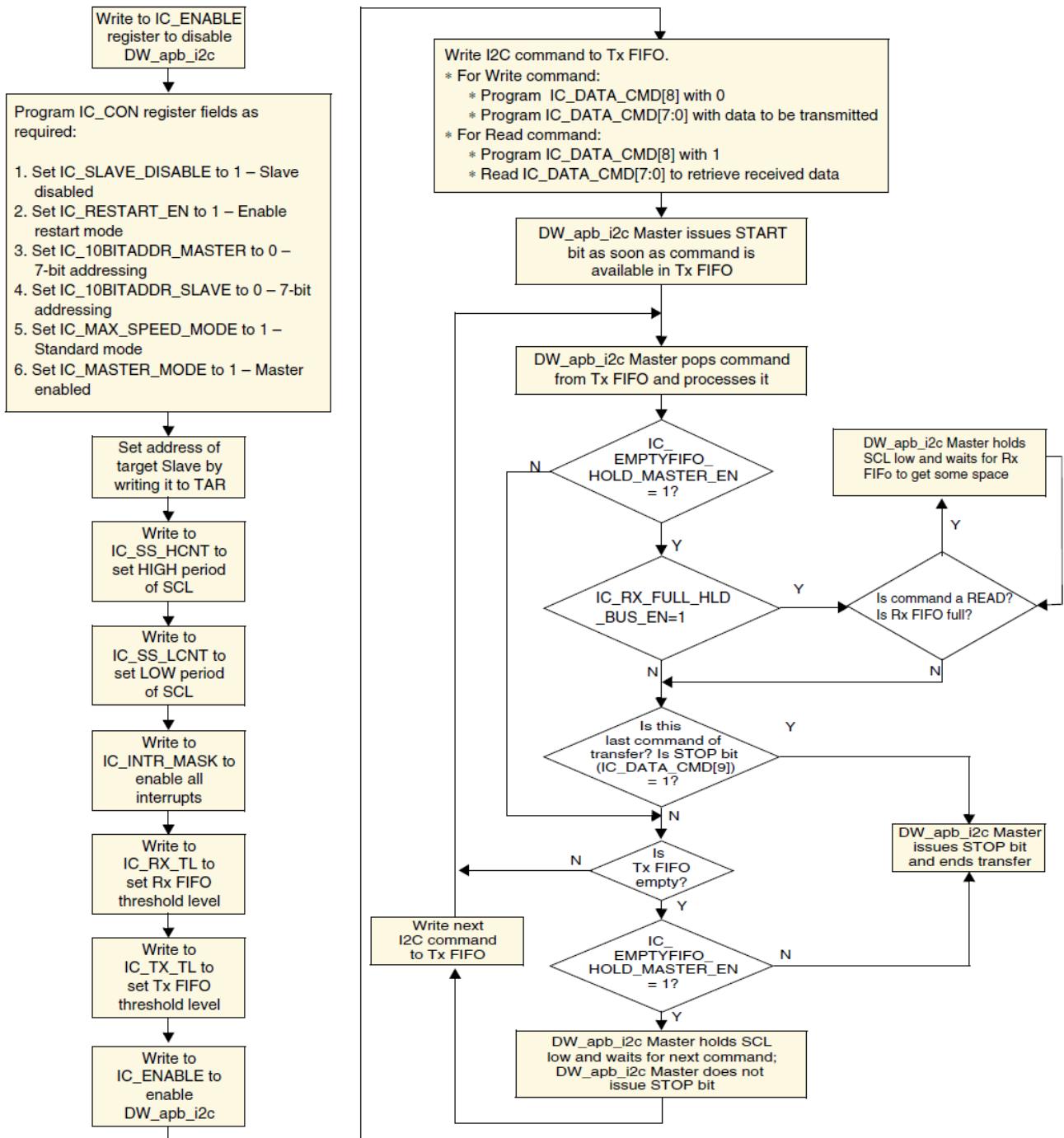


Figure 15.56: Flowchart for DW\_apb\_i2c Master

The flow diagram in Figure 15.57 shows a programming example for the DW\_apb\_i2c master in standard mode, fast mode, or fast mode plus with 7-bit addressing.



**Figure 15.57:** Flowchart for DW\_apb\_i2c Master in Standard Mode, Fast Mode, or Fast Mode Plus

The flow diagram in Figure 15.58 shows a programming example for DW\_apb\_i2c as master with TAR address update. This flow shows how the MST\_ON\_HOLD interrupt is used when the software needs information from the hardware to safely update the TAR address.

## Note

When the software has full knowledge of when it is safe to update the TAR address without requiring information from hardware, the MST\_ON\_HOLD interrupt is not required to update the TAR address.

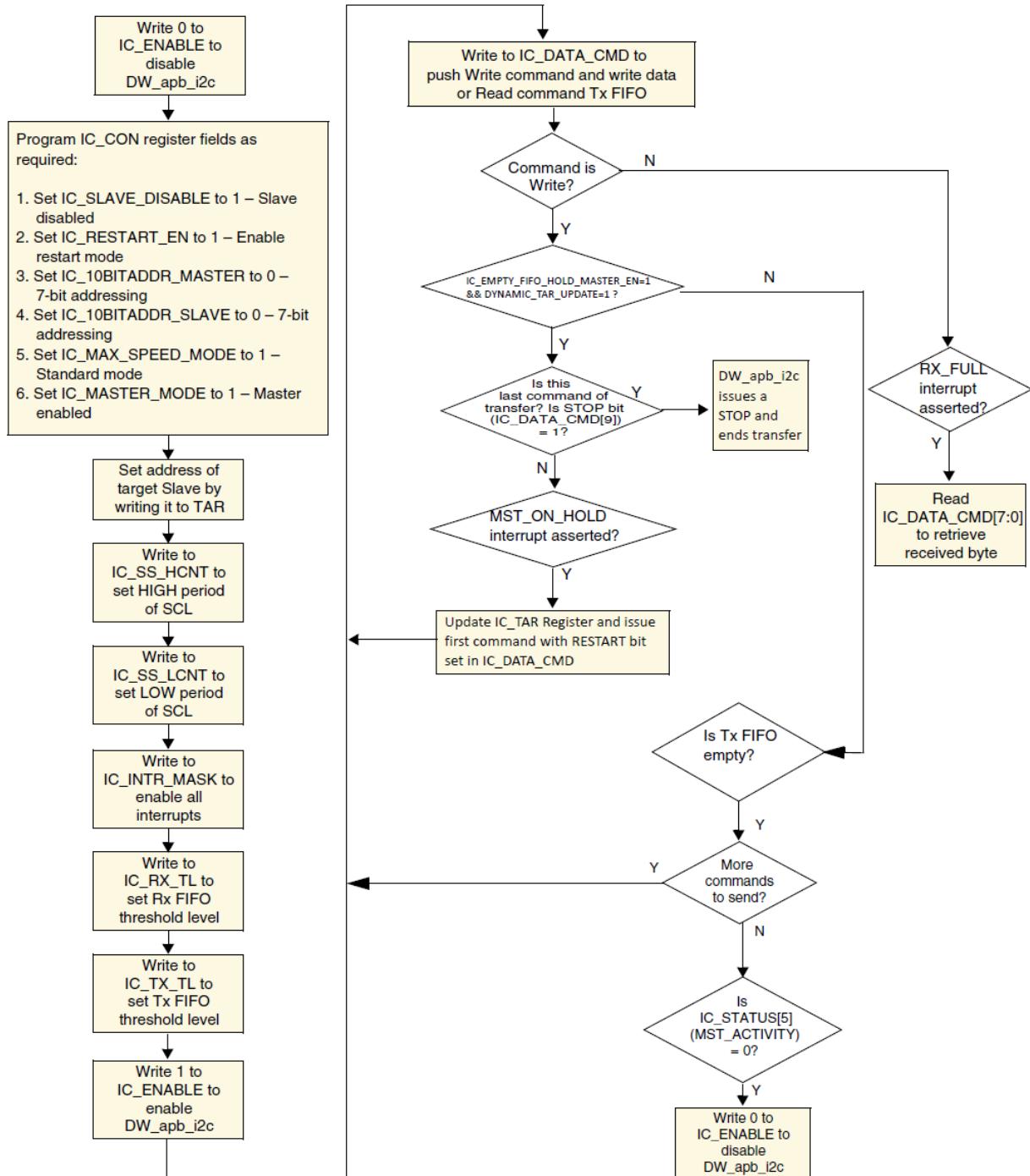
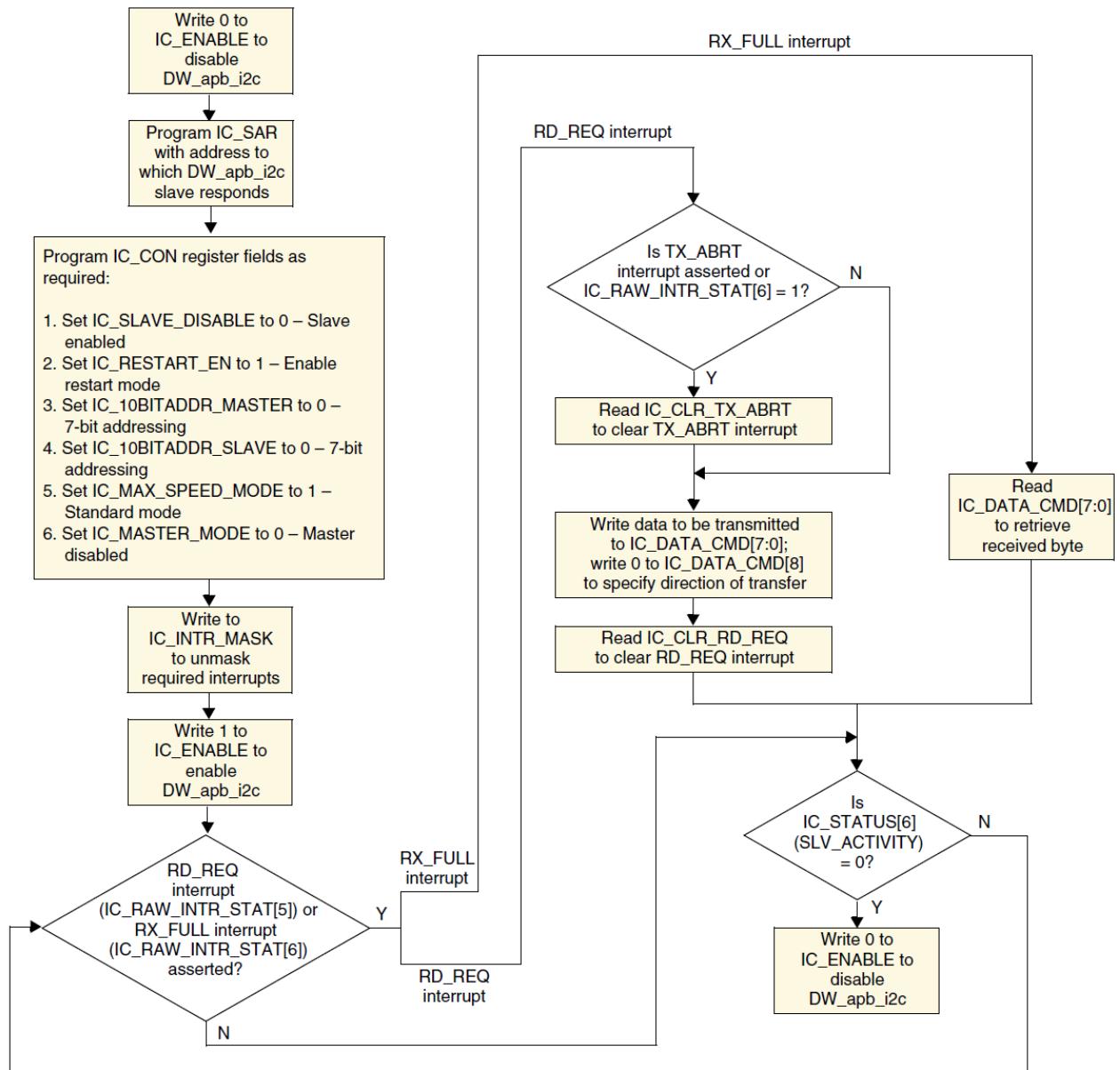


Figure 15.58: Flowchart for DW\_apb\_i2c Master with TAR Address Update

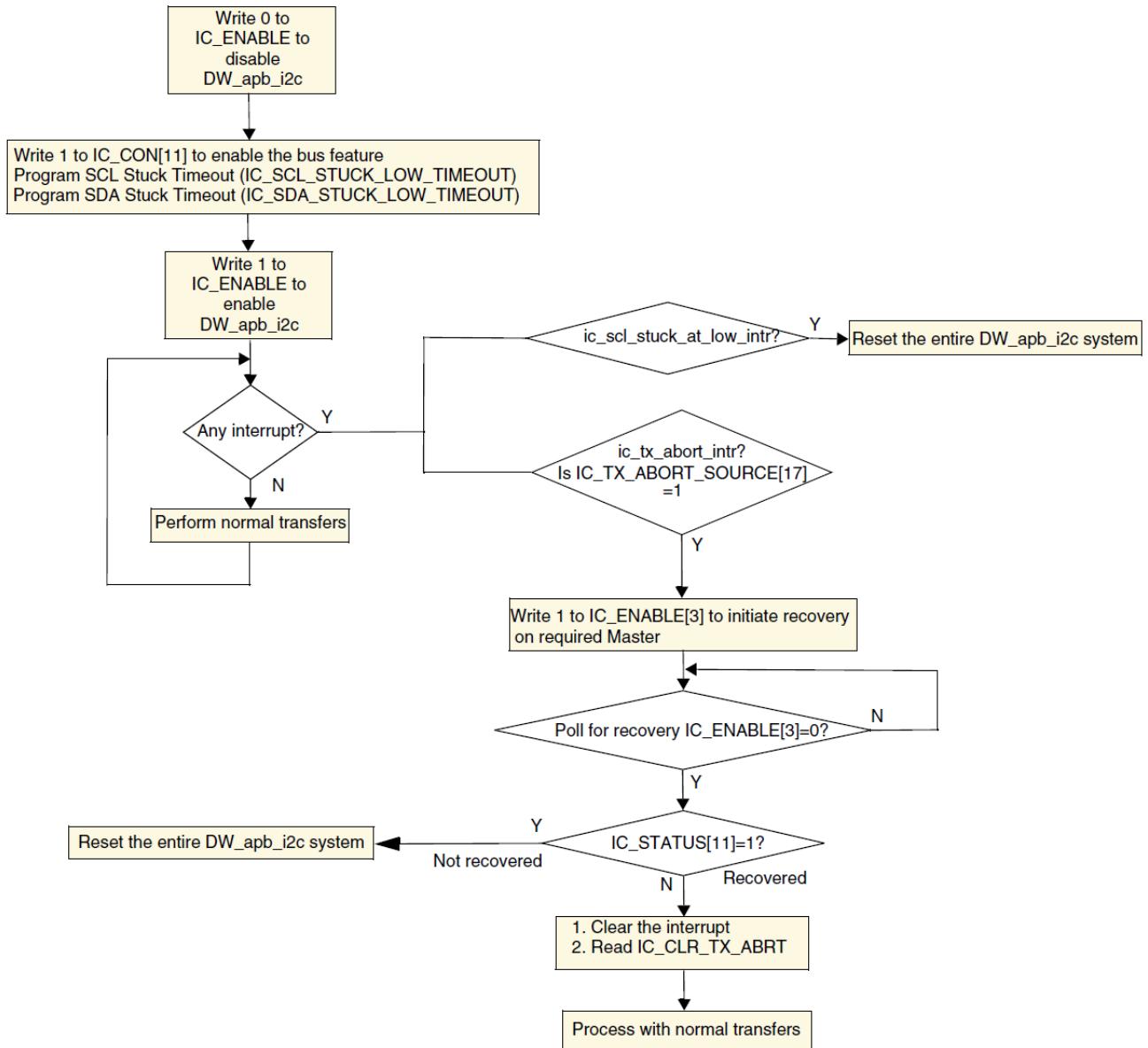
The flow diagram in Figure 15.59 shows a programming example for the DW\_apb\_i2c Slave in standard mode, fast mode, or fast mode plus with 7-bit addressing.



**Figure 15.59:** Flowchart for DW\_apb\_i2c Slave in Standard Mode, Fast Mode, or Fast Mode Plus with 7-bit Addressing

### 15.6.4 Programming Flow for SCL and SDA Bus Recovery

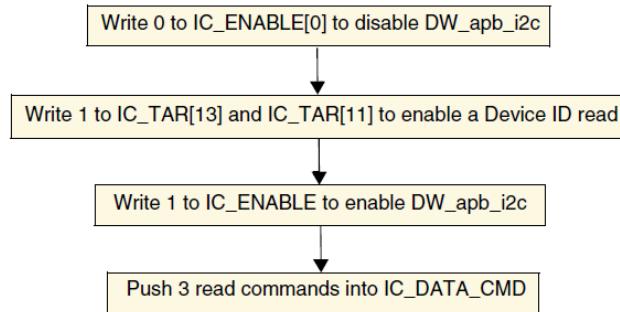
The flow diagram in Figure 15.60 shows a programming example for SCL and SDA bus recovery.



**Figure 15.60:** Flowchart for SCL and SDA Bus Recovery

### 15.6.5 Programming Flow for Reading the Device ID

Figure 15.61 shows a programming flow in the master to initiate a Device ID read.

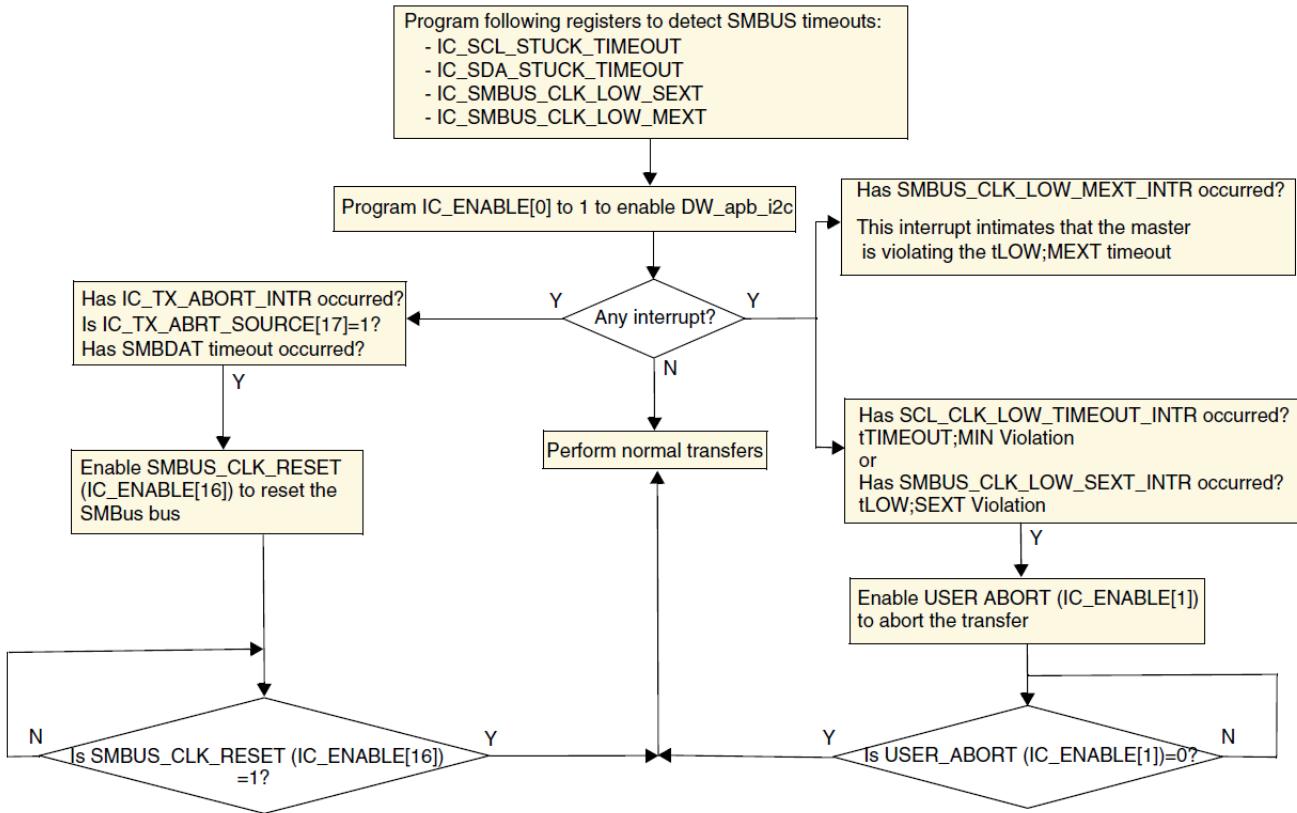


**Figure 15.61:** Flowchart for Reading a Device ID

As the Device ID consists of 3 bytes, the user must issue 3 read commands in IC\_DATA\_CMD register. One read command populates one byte of Device ID in RX FIFO. If more than 3 commands are issued, the Device ID will roll back.

### 15.6.6 Programming Flow for SMBUS Timeout in Master Mode

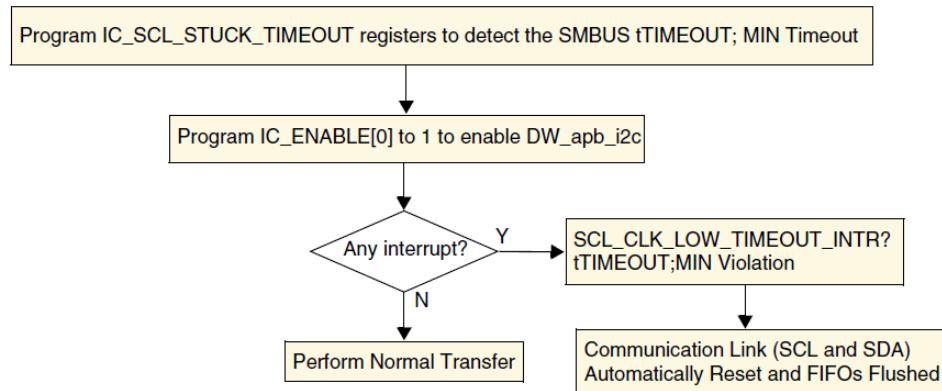
Figure 15.62 shows a programming flow for SMBus timeout in master mode.



**Figure 15.62:** SMBUS Timeout Programming Flow in Master Mode

### 15.6.7 Programming Flow for SMBUS Timeout in Slave Mode

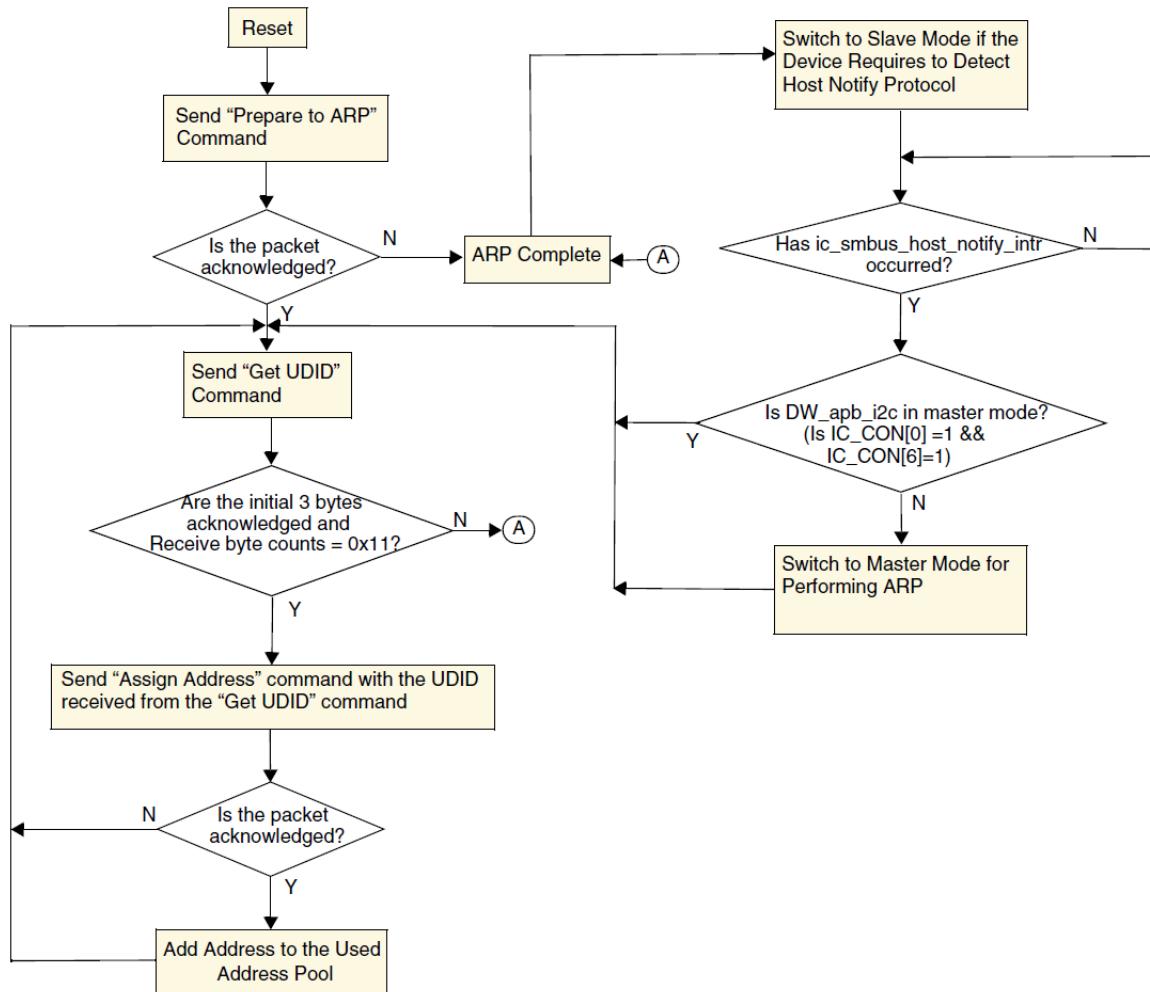
Figure 15.63 shows a programming flow for SMBus timeout in slave mode.



**Figure 15.63:** SMBUS Timeout Programming Flow in Slave Mode

### 15.6.8 ARP Master Programming Flow

Figure 15.64 shows the programming flow for an ARP master.



**Figure 15.64:** ARP Master Programming Flow

### 15.6.9 ARP Slave Programming Flow

Figure 15.65 shows the programming flow for an ARP slave.

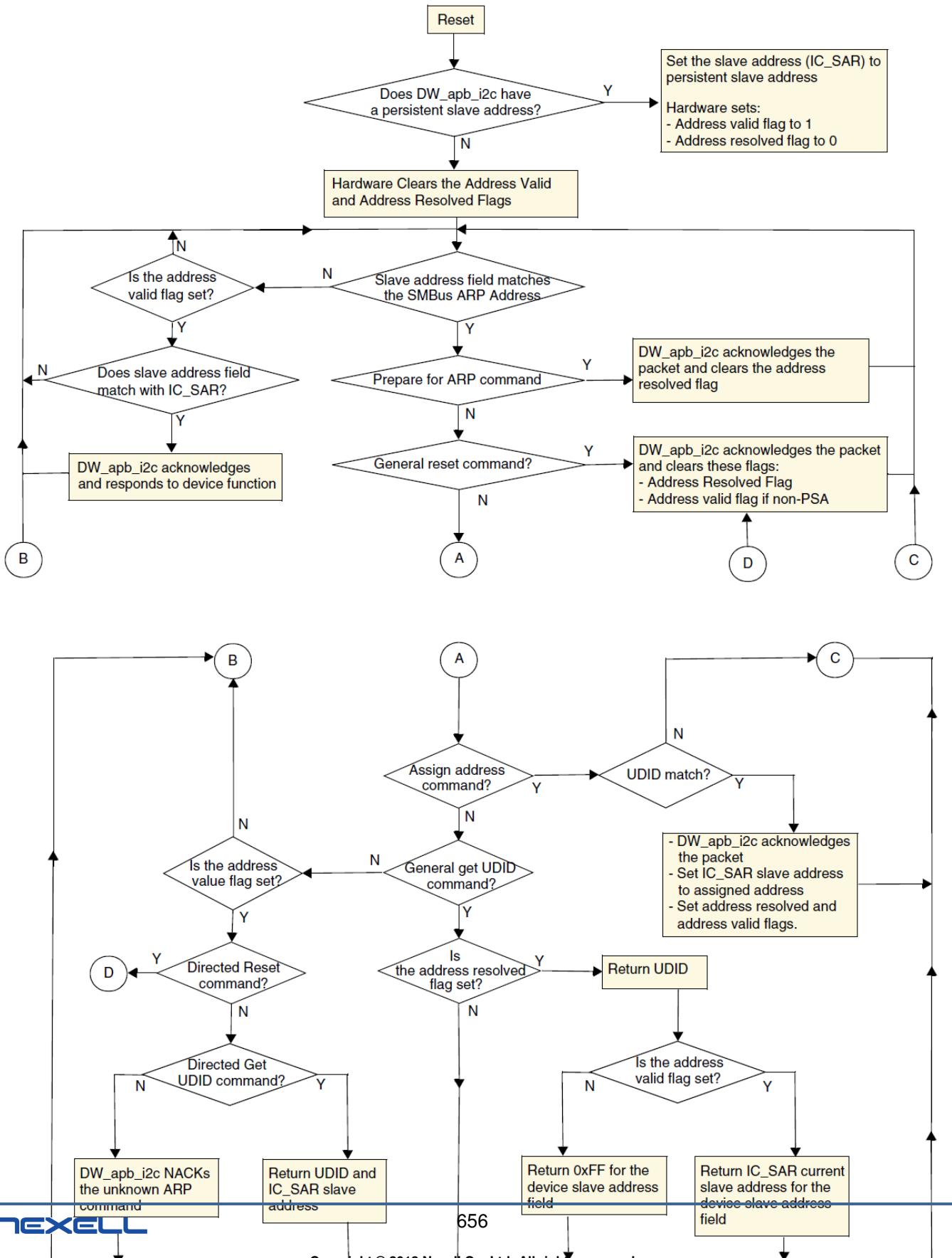


Figure 15.65: ARP Slave Programming Flow

### 15.6.10 SMBus SUSPEND Programming Flow in Host Mode

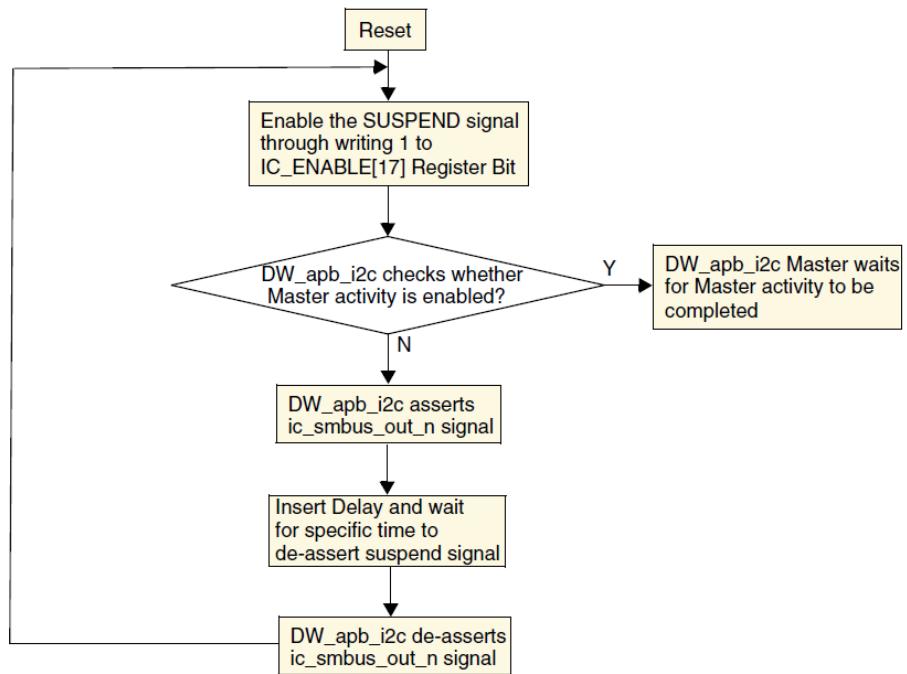
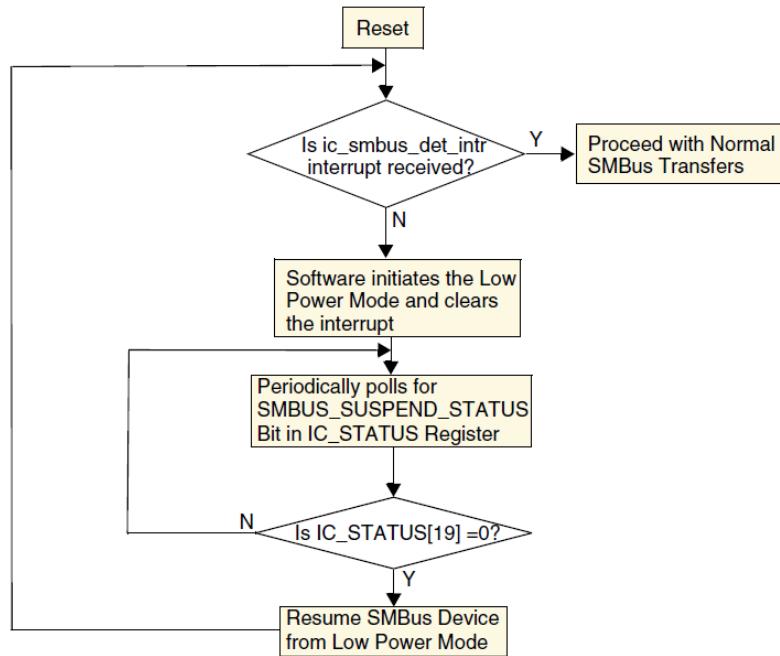


Figure 15.66: Suspend Programming Flow in Host Mode

### 15.6.11 SMBus SUSPEND Programming Flow in Device Mode



**Figure 15.67:** SMBus SUSPEND Programming flow in Device Mode

### 15.6.12 SMBus ALERT Programming Flow in Host Mode

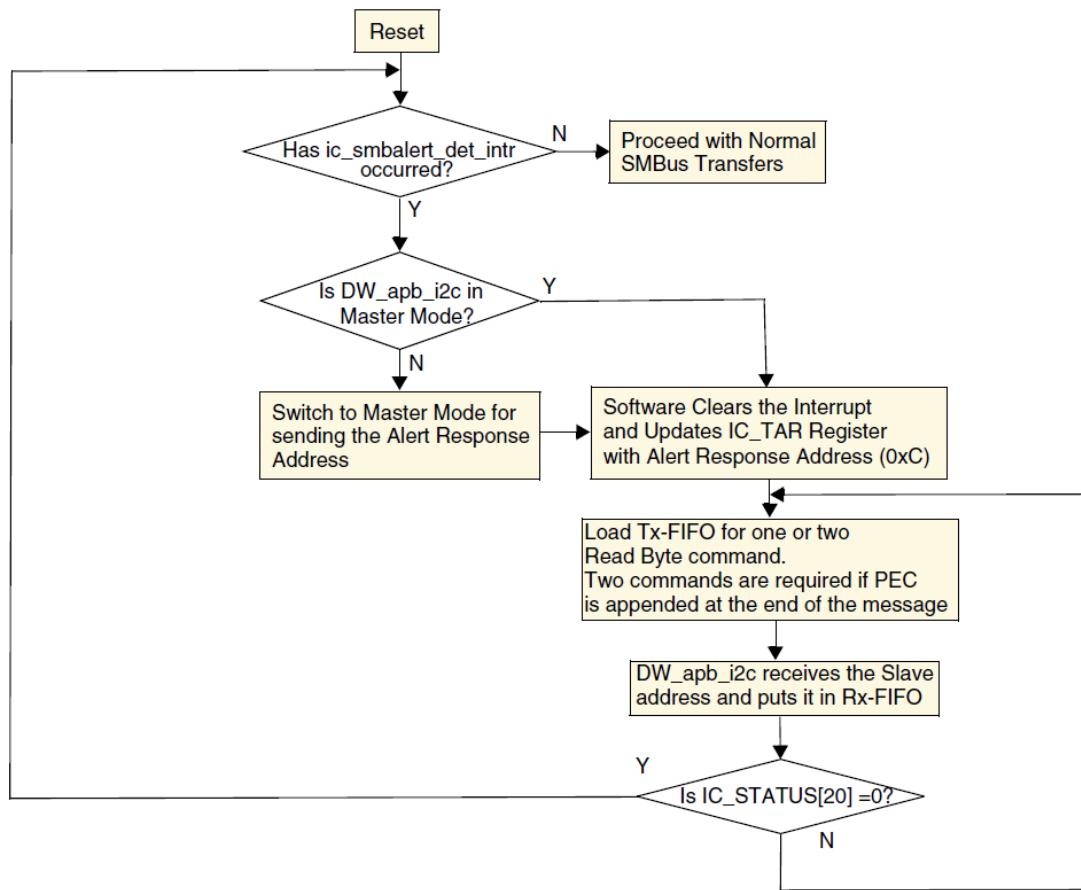
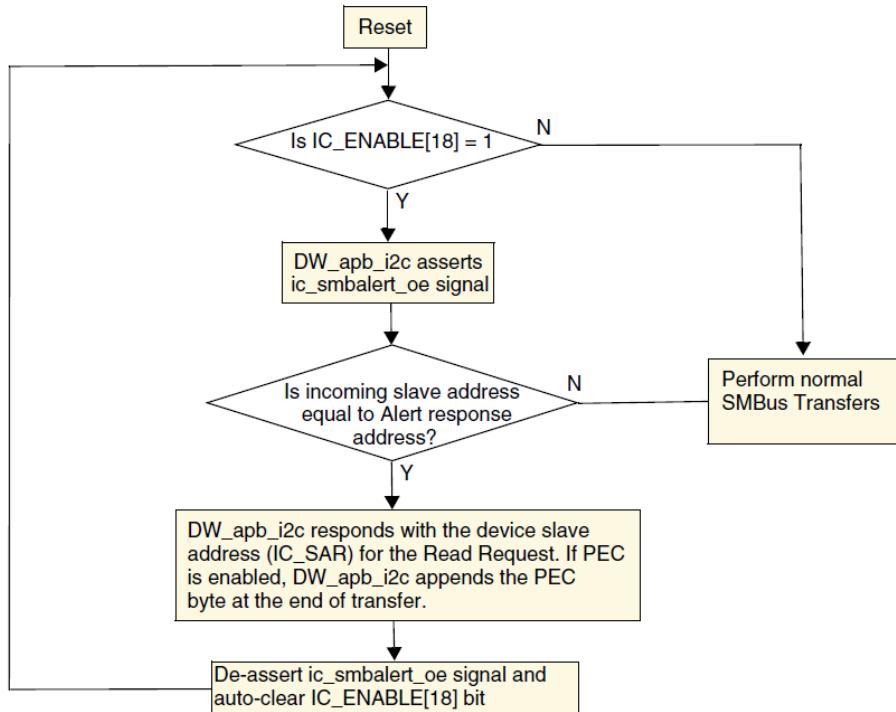


Figure 15.68: SMBus Alert Programming Flow in Host Mode

### 15.6.13 SMBus ALERT Programming Flow in Device Mode



**Figure 15.69:** SMBus Alert Programming Flow in Device Mode

## 15.6.14 Programming Flow Of DW\_apb\_i2c in Ultra-Fast Mode

### 15.6.14.1 DW\_apb\_i2c Master Mode

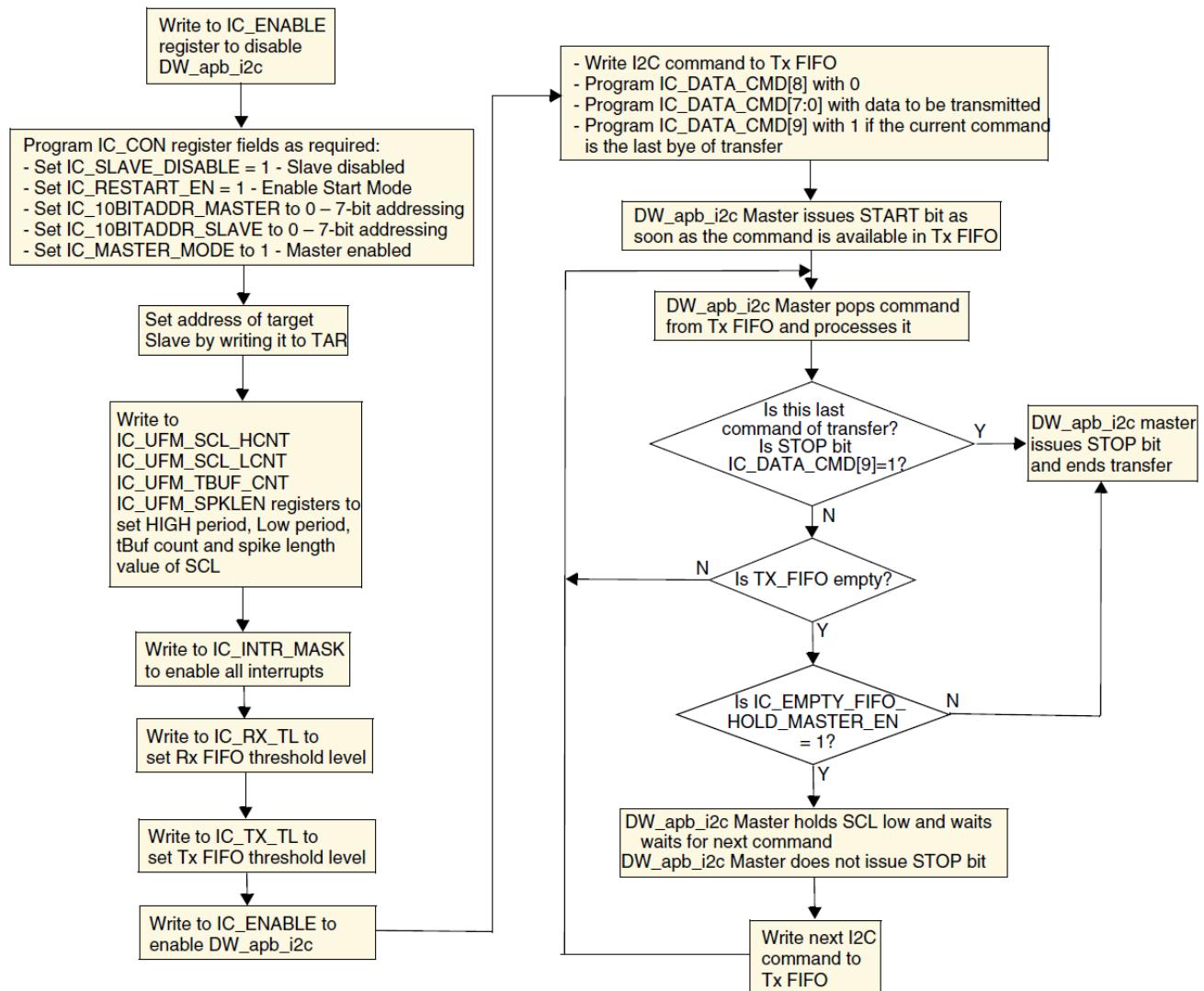


Figure 15.70: DW\_apb\_i2c Ultra-Fast Master Mode

### 15.6.14.2 DW\_apb\_i2c Slave Mode

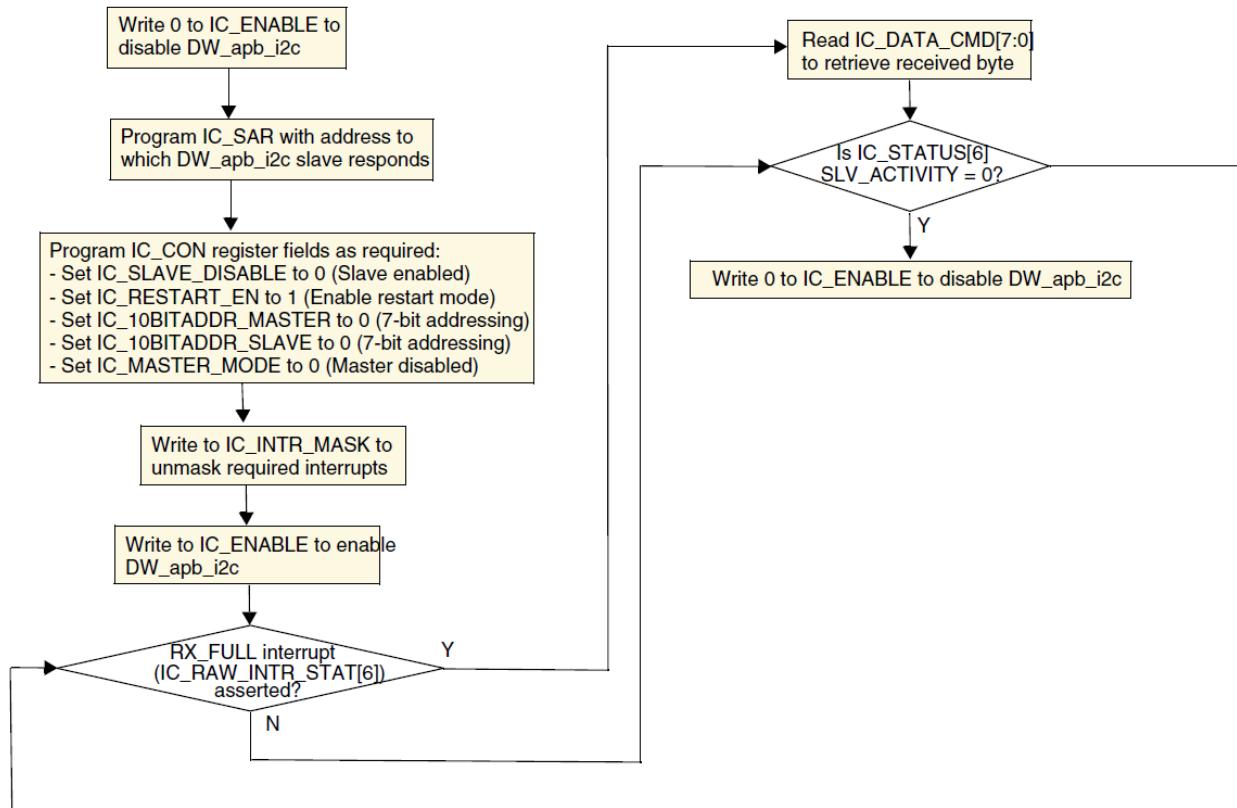


Figure 15.71: DW\_apb\_i2c Ultra-Fast Slave Mode

# 16 SDMMC

## 16.1 Overview

Mobile Storage Host is an interface for Secure Digital Multimedia Card controller, which simultaneously supports Secure Digital memory (SD Mem), Secure Digital I/O (SDIO), Multimedia Cards (MMC), and Consumer Electronics Advanced Transport Architecture (CE-ATA).

Mobile Storage Host specifications are:

- SD Memory Card Specification, Version 3.01
- Secure Digital I/O (SDIO - version 3.0)
- Consumer Electronics Advanced Transport Architecture (CE-ATA - version 1.1)
- Multimedia Cards (MMC - version 4.41, eMMC 4.51)

## 16.2 Features

The key features of Mobile Storage Host support:

- Secure digital memory protocol commands
- Secure digital I/O protocol commands
- MMC protocol commands
- CE-ATA digital protocol commands
- Command completion signal and interrupt to host processor
- Command completion signal disable feature
- FIFO SIZE is 1024byte

The following features of MMC 4.41 are supported:

- DDR in 4-bit and 8-bit mode
- GO\_PRE\_IDLE\_STATE command (CMD0 with argument F0F0F0F0)
- New EXTCSD registers
- Hardware Reset as supported by eMMC 4.41

The following IP specific features of eMMC 4.5 are supported:

- Packed Commands, CMD21, CMD49

- Support for 1.8/3.3V of operation control
- START bit behavior change for DDR modes

The following features of eMMC 4.41 and higher protocol versions are not supported:

- Boot in DDR mode
- Support for 1.2V of operation control

The following features of eMMC 4.5 and higher protocol versions are not supported:

- HS200 mode

## 16.3 Block Diagram

Two main functional blocks of Mobile Storage Host are:

- Bus Interface Unit (BIU) - provides AMBA AHB/APB and DMA interfaces for register and data Read/Write
- Card Interface Unit (CIU) - manages the SD\_MMC\_CEATA protocols and provides clock management

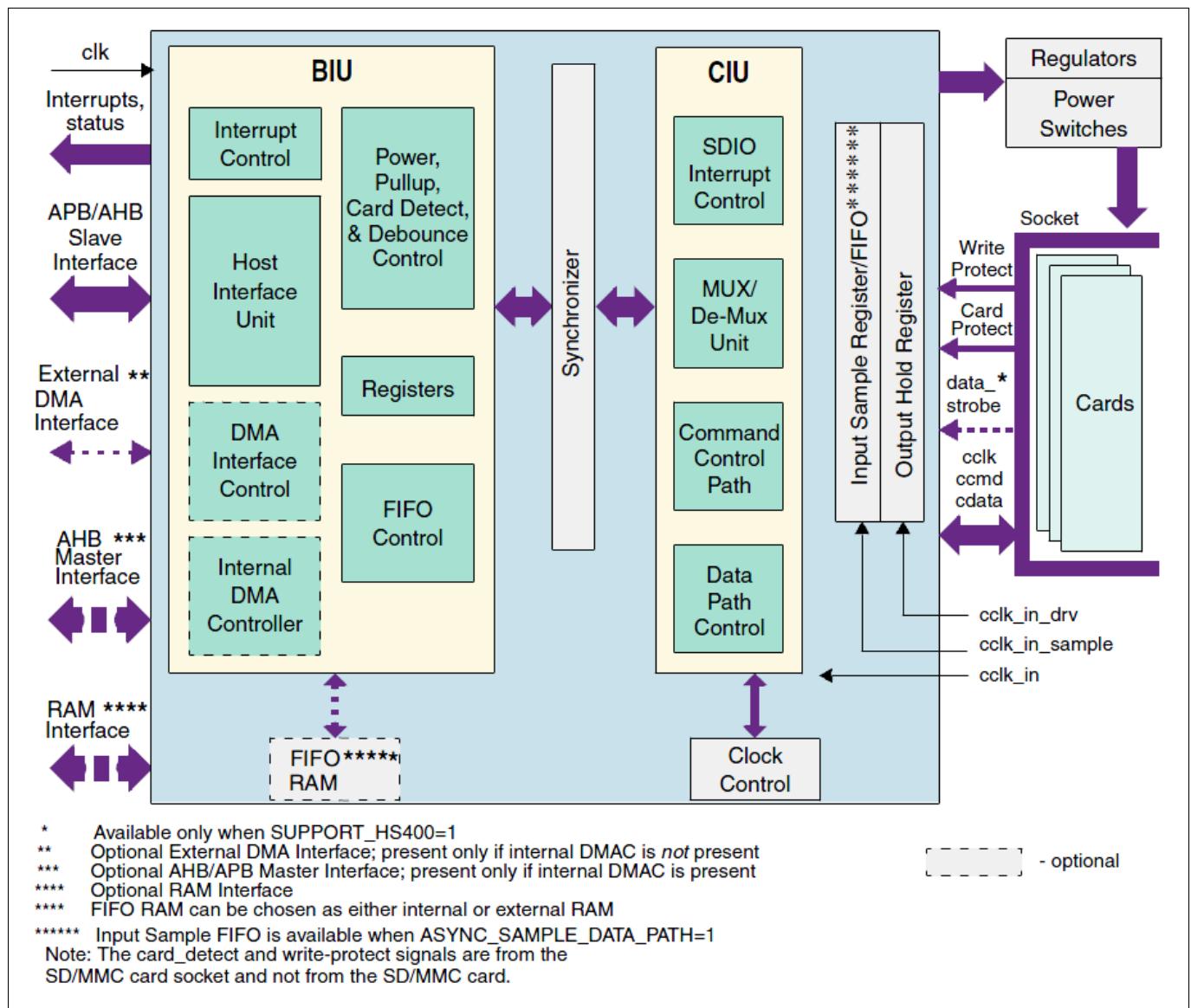


Figure 16.1: Block Diagram of Mobile Storage Host

The BIU provides the host interface to the registers and data FIFO through the Host Interface Unit (HIU). Additionally, it provides an independent data FIFO access through a DMA interface.

The Internal DMA Controller (IDMAC) exchanges data between FIFO and host memory. The host can access a set of IDMAC registers for controlling the IDMAC operation through the AMBA AHB/APB Slave interface.

Mobile Storage Host CIU controls the card-specific protocols. Within the CIU, the command path control unit and the data path control unit interface with the controller to the command and data ports of the SD\_MMC\_CEATA cards. The CIU also provides a clock control.

## 16.4 Clock Logic

SD/MMC card receives DATA/CMD with card clock from the host controller. To synchronize the clock and DATA/CMD, insert the clock delay into the TX/RX clock path by shifting phases and fine-tuning the appropriate clock. To achieve this, the design is provided with additional logic (refer to Figure 16.2 for more information). You can achieve clock selection by using the TIE\_DRV\_SHIFT, TIE\_SMP\_SHIFT register as described at the end of this chapter. ACLK is used for bus operations only.

Figure 16.2 illustrates the clock phase shifter.

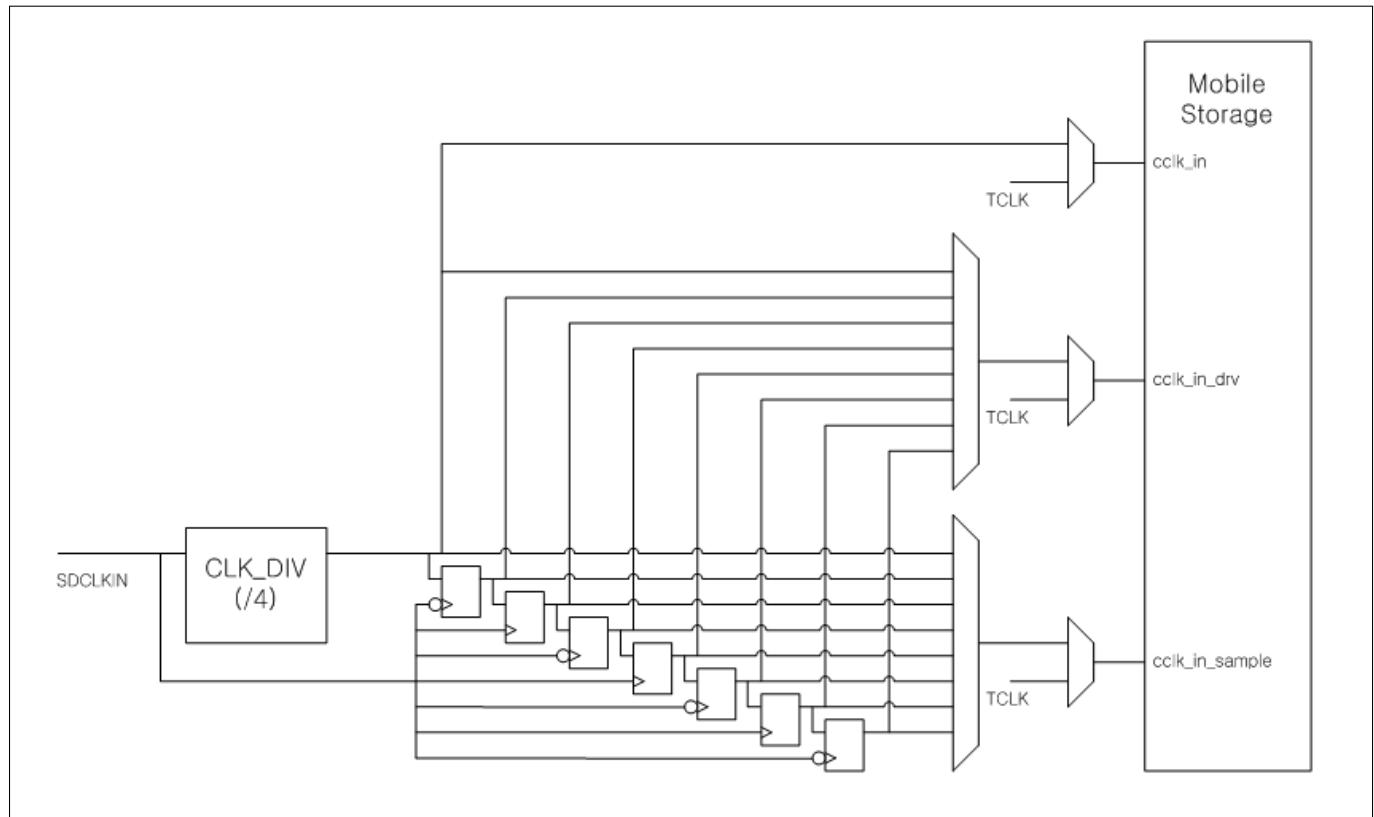


Figure 16.2: Clock Phase Shifter

This clock phase shifter makes 0 degree, 45 degree, 90 degree, 135 degree, 180 degree, 225 degree, 270 degree, and 315 degree phase shifted clocks for TX/RX respectively. To generate 200 MHz phase shifted clock, SDCLKIN should be 800 MHz.

## 16.5 Programmer's Guide

This section includes:

- General restrictions
- Clock programming
- Sending a command
- Command response
- IDMAC descriptor

### 16.5.1 General Restrictions

Issue a data transfer command one time.

The software should ensure that:

- The card is not busy due to any previous command.
- There is no data or command transfer in progress before changing the card clock frequency.

### 16.5.2 Clock Programming

This section includes:

- Enable and disable clock
- CIU update

#### 16.5.2.1 Enable and Disable Clock

The software enables the clock enable bit by using the CMD register.

The update sequence is:

- The software ensures that the card is not busy due to any previous data command.
- The software sets the clock enable or disable register (CLKENA) to one.
- The software sends a command to the CIU to update clock registers by setting the CMD register (refer to CMD for more information).
  - start\_cmd bit
  - update\_clock\_registers\_only
  - wait\_prvdata\_complete
- The software waits till the start\_cmd bit becomes 0.

### 16.5.2.2 CIU Update

The software should stop the clock while it updates the clock divider and clock source registers to prevent clock glitch.

The sequence for updating the clock divider setting is:

1. The software disables the clock.
2. The software updates the clock divider and/or the clock source registers.
3. The software enables the clock.

### 16.5.3 Sending a Command

Four types of command for SD/MMC card are:

- Broadcast commands (bc) with no response
- Broadcast commands with response (bcr)
- Addressed (point-to-point) commands (ac) with no data transfer on DAT lines
- Addressed (point-to-point) data transfer commands (adtc) with no data transfer on DAT lines

### 16.5.4 Command Response

Software programs CMDARG and CMD registers to send a command without data transfer. RESP0 register stores a short response. RESP0, RESP1, RESP2 and RESP3 registers store a long response.

When the MSH receives a response, the command done bit in the RINTSTS register is set as one.

### 16.5.5 IDMAC Descriptor

The IDMAC uses these types of descriptor structures:

- Dual-Buffer Structure - The distance between two descriptors is determined by the Skip Length value programmed in the Descriptor Skip Length (DSL) field of the Bus Mode Register (BMOD @0x80).
- Chain Structure - Each descriptor points to a unique buffer and the next descriptor.

Name	Bit	Description
OWN	[31]	When set, this bit indicates that the descriptor the IDMAC owns. When this bit is reset, it indicates that the descriptor the host owns. The IDMAC clears this bit when it completes the data transfer
Card Error Summary(CES)	[30]	These error bits indicate the status of the transactions to or from the card. These bits are also present in RINTSTS. Indicates the logical OR of the following bits: EBE: End bit Error RTO: Response Time out RCRC: Response CRC SBE: Start Bit Error DRTO: Data Read Timeout DCRC: Data CRC for Receive RE: Response Error
RSVD	[29:6]	Reserved
End of Ring(ER)	[5]	When set, this bit indicates that the descriptor list reached its final descriptor. The IDMAC returns to the base address of the list, creating a descriptor ring. This is meaningful for only a dual-buffer descriptor structure.
Second Address Chained(CH)	[4]	When set, this bit indicates that the second address in the descriptor is the next descriptor address rather than the second buffer address. When this bit is set, BS2 (DES1[25:13]) should be all zeros.
First Descriptor(FS)	[3]	When set, this bit indicates that this descriptor contains the first buffer of the data. If the size of the first buffer is 0, next descriptor contains the beginning of the data.
Last Descriptor(LS)	[2]	This bit is associated with the last block of a DMA transfer. When set, the bit indicates that the buffers pointed to by this descriptor are the last buffers of the data. After this descriptor is completed, the remaining byte count is 0. Alternatively, after the descriptor with the LD bit set is completed, the remaining byte count should be 0.
Disable Interrupt on Completion(DIC)	[1]	When set, this bit will prevent the setting of the TI/RI bit of the IDMAC status register (IDSTS) for the data that ends in the buffer pointed to by this descriptor.
RSVD	[0]	Reserved

Table 16.1: DES0

Name	Bit	Description
RSVD	[31:0]	Reserved

Table 16.2: DES1

Name	Bit	Description
RSVD	[31:25]	Reserved
Buffer 2 Size (BS2)	[24:13]	<p>These bits indicate the second data buffer byte size. The buffer size must be a multiple of 2, 4, or 8, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and proceeds to the next buffer in case of a dual-buffer structure.</p> <p>This field is not valid for chain structure; that is, if DES0[4] is set. If Buffer 2 Size is set to 0 in any of the descriptor, the remaining descriptor cannot have non zero value for Buffer 2 size until the end of the descriptor.</p>
Buffer 1 Size (BS1)	[12:0]	<p>Indicates the data buffer byte size, which must be a multiple of 2, 4, or 8 bytes, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. This field should not be zero.</p> <p>Note: If there is only one buffer to be programmed, you need to use only the Buffer 1, and not Buffer 2.</p>

**Table 16.3:** DES2

Name	Bit	Description
RSVD	[31:0]	Reserved

**Table 16.4:** DES3

Name	Bit	Description
Lower 32-bits of Buffer Address Pointer 1 (BAP1)	[31:0]	These bits indicate the lower 32-bits of the 64-bit physical address of the first data buffer. The IDMAC ignores DES4 [2/1:0:0], corresponding to the bus width of 64/32/16, internally.

**Table 16.5:** DES4

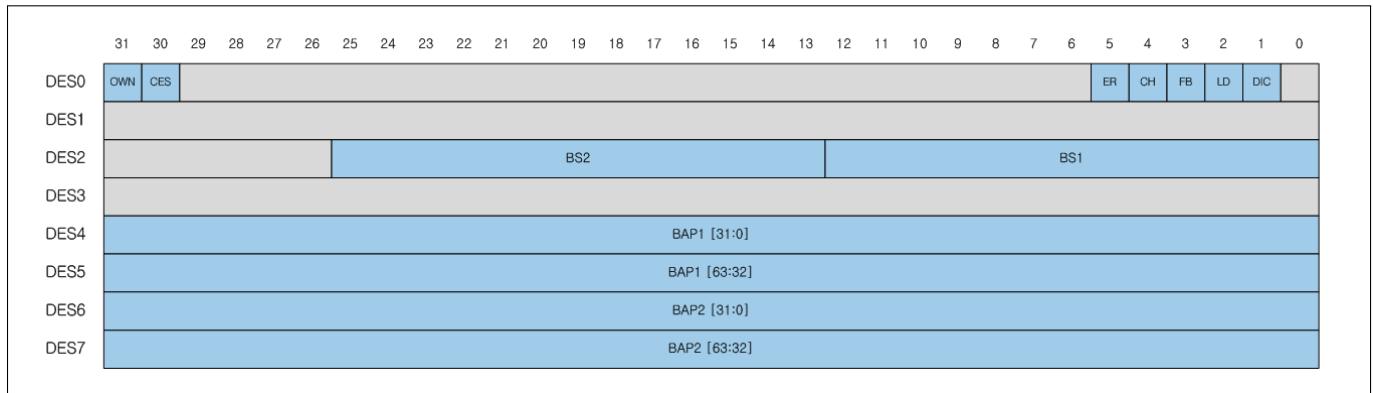
Name	Bit	Description
Upper 32-bits of Buffer Address Pointer 1 (BAP1)	[31:0]	These bits indicate the Upper 32-bits of the 64-bit physical address of the first data buffer.

**Table 16.6:** DES5

Name	Bit	Description
Lower 32-bits of Buffer Address Pointer 2/Next Descriptor Address (BAP2)	[31:0]	<p>These bits indicate the upper 32-bits of the 64-bit physical address of the second buffer when the dual-buffer structure is used. If the Second Address Chained (DES0[4]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If this is not the last descriptor, then the Next Descriptor address pointer must be bus-width aligned (DES6[2/1:0:0] = 0 corresponding to bus-width of 64/32/16, Internally the LSBs are ignored).</p>

**Table 16.7:** DES6

Name	Bit	Description
Upper 32-bits of Buffer Address Pointer 2/Next Descriptor Address (BAP2)	[31:0]	These bits indicate the upper 32-bits of the 64-bit physical address of the second buffer when the dual-buffer structure is used. If the Second Address Chained (DES0[4]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present.

**Table 16.8:** DES7**Figure 16.3:** IDMAC Descriptor

### 16.5.6 Register Map Summary

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)

Register	Offset	Description	Reset Value
CTRL	0x00	Control register	0x0000_0000
PWREN	0x04	Power Enable Register	0x0000_0000
CLKDIV	0x08	Clock Divider Register	0x0000_0000
CLKSRC	0x0C	SD Clock Source Register	0x0000_0000
CLKENA	0x10	Clock Enable Register	0x0000_0000
TMOUT	0x14	Timeout Register	0xFFFF_FFO0
CTYPE	0x18	Card Type Register	0x0000_0000
BLKSIZ	0x1C	Block Size Register	0x0000_0200
BYTCNT	0x20	Byte Count Register	0x0000_0000
INTMASK	0x24	Interrupt Mask Register	0x0000_0000
CMDARG	0x28	Command Argument Register	0x0000_0000
CMD	0x2C	Command Register	0x2000_0000
RESP0	0x30	Response Register 0	0x0000_0000
RESP1	0x34	Response Register 1	0x0000_0000
RESP2	0x38	Response Register 2	0x0000_0000
RESP3	0x3C	Response Register 3	0x0000_0000
MINTSTS	0x40	Masked Interrupt Status Register	0x0000_0000
RINTSTS	0x44	Raw Interrupt Status Register	0x0000_0000
STATUS	0x48	Status Register	0x0000_0406
FIFOTH	0x4C	FIFO Threshold Watermark Register	0x0000_0000
CDETECT	0x50	Card Detect Register	0x0000_0000
WRTPRT	0x54	Write Protect Register	0x0000_0000
GPIO	0x58	General Purpose Input/Output Register	0x0000_0000
TCBCNT	0x5C	Transferred CIU Card Byte Count Register	0x0000_0000
TBBCNT	0x60	Transferred Host to BIU-FIFO Byte Count Register	0x0000_0000
DEBNCE	0x64	Debounce Count Register	0x00FF_FFFF
USRID	0x68	User ID Register	0x00DC_2900
VERID	0x6C	Version ID Register	0x5342_270A
HCON	0x70	Hardware Configuration Register	Undefined
UHS_REG	0x74	UHS-1 Register	0x0000_0000
RST_n	0x78	H/W Reset	0x0000_0001
BMOD	0x80	Bus Mode Register	0x0000_0000
PLDMND	0x84	Poll Demand Register	0x0000_0000
DBADDRL	0x88	Descriptor List Base Address Lower Register	0x0000_0000
DBADDRU	0x8C	Descriptor List Base Address Upper Register	0x0000_0000
IDSTS	0x90	Internal DMAC Status Register	0x0000_0000
IDINTEN	0x94	Internal DMAC Interrupt Enable Register	0x0000_0000
DSCADDRL	0x98	Current Host Descriptor Address Lower Register	0x0000_0000
DSCADDRU	0x9C	Current Host Descriptor Address Upper Register	0x0000_0000
BUFADDRL	0xA0	Current Buffer Descriptor Lower Address Register	0x0000_0000
BUFADDRU	0xA4	Current Buffer Descriptor Upper Address Register	0x0000_0000
CardThrCtl	0x100	Card Threshold Control Register	0x0000_0000
Back_end_power	0x104	Back-end Power Register	0x0000_0000
EMMC_DDR_REG	0x10C	eMMC DDR Register	0x0000_0000
ENABLE_SHIFT	0x110	Enable Phase Shift Register	0x0000_0000
TIEOFF_MODE	0x400	Tie-off Mode Register	0x0000_0000
TIEOFF_SRAM	0x404	Tie-off SRAM Register	0x0000_0000
TIEOFF_DRV_PHASE	0x408	Tie-off Drive Phase Register	0x0000_0000
TIEOFF_SMP_PHASE	0x40C	Tie-off Sample Phase Register	0x0000_0000

TIEOFF_DS_DELAY	0x410	Tie-off Data Strobe Delay Register	0x0000_0040
-----------------	-------	------------------------------------	-------------

### 16.5.7 CTRL

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:26]	-	Reserved	-
use_internal_dmac	[25]	R/W	Present only for the Internal DMAC configuration; else, it is reserved. 0 : The host performs data transfers through the slave interface 1 : Internal DMAC used for data transfer	0x0
enable_OD_pullup	[24]	R/W	External open-drain pullup: 0 : Disable 1 : Enable Inverted value of this bit is output to ccmd_od_pullup_en_n port. When bit is set, command output always driven in open-drive mode; that is, DWC_mobile_storage drives either 0 or high impedance, and does not drive hard 1.	0x0
Card_voltage_b	[23:20]	R/W	Card regulator-B voltage setting; output to card_volt_b port. Optional feature; ports can be used as general-purpose outputs.	0x0
Card_voltage_a	[19:16]	R/W	Card regulator-A voltage setting; output to card_volt_a port. Optional feature; ports can be used as general-purpose outputs.	0x0
RSVD	[15:12]	-	Reserved	-
ceata_device_interrupt_status	[11]	R/W	0 : Interrupts not enabled in CE-ATA device (nIEN = 1 in ATA control register) 1 : Interrupts are enabled in CE-ATA device (nIEN = 0 in ATA control register) Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled (nIEN = 1). If the host enables CE-ATA device interrupt, then software should set this bit.	0x0
send_auto_stop_ccsd	[10]	R/W	0 : Clear bit if DWC_mobile_storage does not reset the bit. 1 : Send internally generated STOP after sending CCSD to CE-ATA device. NOTE: Always set send_auto_stop_ccsd and send_ccsd bits together; send_auto_stop_ccsd should not be set independent of send_ccsd. When set, DWC_Mobile_Storage automatically sends internally-generated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in RINTSTS is set and generates interrupt to host if Auto Command Done interrupt is not masked. After sending the CCSD, DWC_mobile_storage automatically clears send_auto_stop_ccsd bit.	0x0
send_ccsd	[9]	R/W	0 : Clear bit if DWC_mobile_storage does not reset the bit. 1 : Send Command Completion Signal Disable (CCSD) to CE-ATA device When set, DWC_mobile_storage sends CCSD to CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device. Once the CCSD pattern is sent to device, DWC_mobile_storage automatically clears send_ccsd bit. It also sets Command Done (CD) bit in RINTSTS register and generates interrupt to host if Command Done interrupt is not masked. NOTE: Once send_ccsd bit is set, it takes two card clock cycles to drive the CCSD on the CMD line. Due to this, during the boundary conditions it may happen that CCSD is sent to the CE-ATA device, even if the device signalled CCS.	0x0
abort_read_data	[8]	R/W	0 : No change 1 : After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data state-machine resets to idle. Used in SDIO card suspend sequence.	0x0

send_irq_response	[7]	R/W	0 : No change 1 : Send auto IRQ response Bit automatically clears once response is sent. To wait for MMC card interrupts, host issues CMD40, and DWC_mobile_storage waits for interrupt response from MMC card(s). In meantime, if host wants DWC_mobile_storage to exit waiting for interrupt state, it can set this bit, at which time DWC_mobile_storage command state-machine sends CMD40 response on bus and returns to idle state.	0x0
read_wait	[6]	R/W	0 : Disable DMA transfer mode 1 : Enable DMA transfer mode Valid only if DWC_mobile_storage configured for External DMA interface.  Even when DMA mode is enabled, host can still push/pop data into or from FIFO; this should not happen during the normal operation. If there is simultaneous FIFO access from host/DMA, the data coherency is lost. Also, there is no arbitration inside DWC_mobile_storage to prioritize simultaneous host/DMA access.	0x0
dma_enable	[5]	R/W	0 : Disable DMA transfer mode 1 : Enable DMA transfer mode Valid only if DWC_mobile_storage configured for External DMA interface.  Even when DMA mode is enabled, host can still push/pop data into or from FIFO; this should not happen during the normal operation. If there is simultaneous FIFO access from host/DMA, the data coherency is lost. Also, there is no arbitration inside DWC_mobile_storage to prioritize simultaneous host/DMA access.	0x0
int_enable	[4]	R/W	Global interrupt enable/disable bit: 0 : Disable interrupts 1 : Enable interrupts The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.	0x0
RSVD	[3]	-	Reserved	-
dma_reset	[2]	R/W	0 : No change 1 : Reset internal DMA interface control logic To reset DMA interface, firmware should set bit to 1. This bit is auto-cleared after two AHB clocks.	0x0
fifo_reset	[1]	R/W	0 : No change 1 : Reset to data FIFO To reset FIFO pointers To reset FIFO, firmware should set bit to 1. This bit is auto-cleared after completion of reset operation. Note: FIFO pointers will be out of reset after 2 cycles of system clocks in addition to synchronization delay (2 cycles of card clock), after the fifo_reset is cleared.	0x0
controller_reset	[0]	R/W	0 : No change 1 : Reset DWC_mobile_storage controller To reset controller, firmware should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets: * BIU/CIU interface * CIU and state machines * abort_read_data, send_irq_response, and read_wait bits of Control register * start_cmd bit of Command register Does not affect any registers or DMA interface, or FIFO or host interrupts	0x0

### 16.5.8 PWREN

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:30]	-	Reserved	-

power_enable	[29:0]	R/W	<p>Power on/off switch for up to 16 cards; for example, bit[0] controls card 0.</p> <p>Once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card.</p> <p>0 : power off 1 : power on</p> <p>Only NUM_CARDS number of bits are implemented.</p> <p>Bit values output to card_power_en port.</p> <p>Optional feature; ports can be used as general-purpose outputs.</p> <p>NOTE: An external circuitry is required to drive the CMD and DAT lines to low during the power off (down) cycle.</p>	0x0
--------------	--------	-----	---	-----

### 16.5.9 CLKDIV

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
clk_divider3	[31:24]	R/W	Clock divider-3 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), a value of 1 means divide by $2^1 = 2$ , a value of $\text{FFFF}$ means divide by $2^{255} = 510$ , and so on.	0x0
clk_divider2	[23:16]	R/W	Clock divider-2 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$ , value of $\text{FFFF}$ means divide by $2^{255} = 510$ , and so on.	0x0
clk_divider1	[15:8]	R/W	Clock divider-1 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$ , value of $\text{FFFF}$ means divide by $2^{255} = 510$ , and so on.	0x0
clk_divider0	[7:0]	R/W	Clock divider-0 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$ , value of $\text{FFFF}$ means divide by $2^{255} = 510$ , and so on.	0x0

### 16.5.10 CLKSRC

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
clk_source	[31:0]	R/W	<p>Clock divider source for up to 16 SD cards supported. Each card has two bits assigned to it. For example, bits[1:0] assigned for card-0, which maps and internally routes clock divider[3:0] outputs to cclk_out[15:0] pins, depending on bit value.</p> <p>00 ? Clock divider 0 01 ? Clock divider 1 10 ? Clock divider 2 11 ? Clock divider 3</p>	0x0

### 16.5.11 CLKENA

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:17]	-	Reserved	-
cclk_low_power	[16]	R/W	Low-power control for up to 16 SD card clocks and one MMC card clock supported. 0 ? Non-low-power mode 1 ? Low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).	0x0
RSVD	[15:1]	-	Reserved	-
cclk_enable	[0]	R/W	Clock-enable control for up to 16 SD card clocks and one MMC card clock supported. 0 ? Clock disabled 1 ? Clock enabled	0x0

### 16.5.12 TMOUT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x14 Reset Value = 0xFFFF\_FF00

Name	Bit	Type	Description	Reset Value
data_timeout	[31:8]	R/W	Value for card Data Read Timeout; same value also used for Data Starvation by Host timeout. The timeout counter is started only after the card clock is stopped. Value is in number of card output clocks ? cclk_out of selected card. Note: The software timer should be used if the timeout value is in the order of 100 ms. In this case, read data timeout interrupt needs to be disabled.	0xFFFFFFFF
response_timeout	[7:0]	R/W	Response timeout value. Value is in number of card output clocks ? cclk_out.	0x0

### 16.5.13 CTYPE

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
card_width	[31:16]	R/W	One bit per card indicates if card is 8-bit: 0 ? Non 8-bit mode 1 ? 8-bit mode Bit[31] corresponds to card[15]; bit[16] corresponds to card[0].	0x0

card_width	[15:0]	R/W	One bit per card indicates if card is 1-bit or 4-bit: 0 ? 1-bit mode 1 ? 4-bit mode Bit[15] corresponds to card[15], bit[0] corresponds to card[0]. Only NUM_CARDS*2 number of bits are implemented.	0x0
------------	--------	-----	--	-----

### 16.5.14 BLKSIZ

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x1C Reset Value = 0x0000\_0200

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
block_size	[15:0]	R/W	Block size	0x200

### 16.5.15 BYTCNT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
byte_count	[31:0]	R/W	Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.	0x0

### 16.5.16 INTMASK

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x24 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
sdio_int_mask	[31:16]	R/W	<p>Mask SDIO interrupts One bit for each card. Bit[31] corresponds to card[15], and bit[16] corresponds to card[0]. When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enables an interrupt. In MMC-Ver3.3-only mode, these bits are always 0.</p>	0x0
int_mask	[15:0]	R/W	<p>Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt. bit 15 ? End-bit error (read)/Write no CRC (EBE) bit 14 ? Auto command done (ACD) bit 13 ? Start Bit Error(SBE)/Busy Clear Interrupt (BCI) bit 12 ? Hardware locked write error (HLE) bit 11 ? FIFO underrun/overrun error (FRUN) bit 10 ? Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9 ? Data read timeout (DRTO) bit 8 ? Response timeout (RTO) bit 7 ? Data CRC error (DCRC) bit 6 ? Response CRC error (RCRC) bit 5 ? Receive FIFO data request (RXDR) bit 4 ? Transmit FIFO data request (TXDR) bit 3 ? Data transfer over (DTO) bit 2 ? Command done (CD) bit 1 ? Response error (RE) bit 0 ? Card detect (CD)</p>	0x0

### 16.5.17 CMDARG

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x28 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
cmd_arg	[31:0]	R/W	Value indicates command argument to be passed to card.	0x0

### 16.5.18 CMD

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x2C Reset Value = 0x2000\_0000

Name	Bit	Type	Description	Reset Value
start_cmd	[31]	R/W	Start command. Once command is taken by CIU, bit is cleared. When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register. Once command is sent and response is received from SD_MMC_CEATA cards, Command Done bit is set in raw interrupt register.	0x0
RSVD	[30]	-	Reserved	-
use_hold_reg	[29]	R/W	Use Hold Register 0 - CMD and DATA sent to card bypassing HOLD Register 1 - CMD and DATA sent to card through the HOLD Register	0x1
volt_switch	[28]	R/W	Voltage switch bit 0 - No voltage switching 1 - Voltage switching enabled; must be set for CMD11 only	0x0
boot_mode	[27]	R/W	Boot Mode 0 - Mandatory Boot operation 1 - Alternate Boot operation	0x0
disable_boot	[26]	R/W	Disable Boot. When software sets this bit along with start_cmd, CIU terminates the boot operation. Do NOT set disable_boot and enable_boot together.	0x0
expect_boot_ack	[25]	R/W	Expect Boot Acknowledge. When Software sets this bit along with enable_boot, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.	0x0
enable_boot	[24]	R/W	Enable Boot?this bit should be set only for mandatory boot mode. When Software sets this bit along with start_cmd, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set disable_boot and enable_boot together.	0x0
ccs_expected	[23]	R/W	0 ? Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register), or command does not expect CCS from device 1 ? Interrupts are enabled in CE-ATA device (nIEN = 0), and RW_BLK command expects command completion signal from CE-ATA device If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. DWC_mobile_storage sets Data Transfer Over (DTO) bit in RINTSTS register and generates interrupt to host if Data Transfer Over interrupt is not masked.	0x0
read_ceata_device	[22]	R/W	0 ? Host is not performing read access (RW_REG or RW_BLK) towards CE-ATA device 1 ? Host is performing read access (RW_REG or RW_BLK) towards CE-ATA device Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data timeout indication while performing CE-ATA read transfers. Maximum value of I/O transmission delay can be no less than 10 seconds. DWC_mobile_storage should not indicate read data timeout while waiting for data from CE-ATA device.	0x0
update_clock_registers_only	[21]	R/W	0 ? Normal command sequence 1 ? Do not send commands, just update clock register value into card clock domain Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when update_clock_registers_only = 0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card(s). When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC_CEATA cards.	0x0

card_number	[20:16]	R/W	<p>Card number in use. Represents physical slot number of card being accessed. In MMC-Ver3.3-only mode, up to 30 cards are supported;</p> <p>in SD-only mode, up to 16 cards are supported. Registered version of this is reflected on dw_dma_card_num and ge_dma_card_num ports, which can be used to create separate DMA requests, if needed.</p> <p>In addition, in SD mode this is used to mux or demux signals from selected card because each card is interfaced to DWC_mobile_storage by separate bus.</p>	0x0
send_initialization	[15]	R/W	<p>0 ? Do not send initialization sequence (80 clocks of 1) before sending this command      1 ? Send initialization sequence before sending this command      After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).</p>	0x0
stop_abort_cmd	[14]	R/W	<p>0 ? Neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0.      1 ? Stop or abort command intended to stop current data transfer in progress.      When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = disable_boot.</p>	0x0
wait_prvdata_complete	[13]	R/W	<p>0 ? Send command at once, even if previous data transfer has not completed      1 ? Wait for previous data transfer completion before sending command      The wait_prvdata_complete = 0 option typically used to query status of card during data transfer or to stop current data transfer; card_number should be same as in previous command.</p>	0x0
send_auto_stop	[12]	R/W	<p>0 ? No stop command sent at end of data transfer      1 ? Send stop command at end of data transfer      When set, DWC_mobile_storage sends stop command to SD_MMC_CEATA cards at end of data transfer.      * when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands      * open-ended transfers that software should explicitly send to stop command      Additionally, when Â“resumeÂ” is sent to resume ? suspended memory access of SD-Combo card ? bit should be set correctly if suspended data transfer needs send_auto_stop.      Don't care if no data expected from card.</p>	0x0
transfer_mode	[11]	R/W	<p>0 ? Block data transfer command      1 ? Stream data transfer command      Don't care if no data expected.</p>	0x0
read/write	[10]	R/W	<p>0 ? Read from card      1 ? Write to card      Don't care if no data expected from card.</p>	0x0
data_expected	[9]	R/W	<p>0 ? No data transfer expected (read/write)      1 ? Data transfer expected (read/write)</p>	0x0
check_response_crc	[8]	R/W	<p>0 ? Do not check response CRC      1 ? Check response CRC      Some of command responses do not return valid CRC bits.      Software should disable CRC checks for those commands in order to disable CRC checking by controller.</p>	0x0
response_length	[7]	R/W	<p>0 ? Short response expected from card      1 ? Long response expected from card</p>	0x0
response_expect	[6]	R/W	<p>0 ? No response expected from card      1 ? Response expected from card</p>	0x0
cmd_index	[5:0]	R/W	Command index	0x0

### 16.5.19 RESP0

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
response 0	[31:0]	R	Bit[31:0] of response	0x0

### 16.5.20 RESP1

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
response 1	[31:0]	R	Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always Á¤ÁíshortÁ¤Áš for them. For information on when CIU sends auto-stop commands	0x0

### 16.5.21 RESP2

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
response 2	[31:0]	R	Bit[95:64] of long response	0x0

### 16.5.22 RESP3

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x3C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
response 3	[31:0]	R	Bit[127:96] of long response	0x0

### 16.5.23 MINTSTS

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
sdio_interrupt	[31:16]	R/W	Interrupt from SDIO card; one bit for each card. Bit[31] corresponds to Card[15], and bit[16] is for Card[0]. SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register (mask bit 1 enables interrupt; 0 masks interrupt). 0 ? No SDIO interrupt from card 1 ? SDIO interrupt from card	0x0
int_status	[15:0]	R/W	Interrupt enabled only if corresponding bit in interrupt mask register is set. bit 15 ? End-bit error (read)/write no CRC (EBE) bit 14 ? Auto command done (ACD) bit 13 ? Start Bit Error(SBE)/Busy Clear Interrupt (BCI) bit 12 ? Hardware locked write error (HLE) bit 11 ? FIFO underrun/overrun error (FRUN) bit 10 ? Data starvation by host timeout (HTO)/Volt_switch_int bit 9 ? Data read timeout (DRTO) bit 8 ? Response timeout (RTO) bit 7 ? Data CRC error (DCRC) bit 6 ? Response CRC error (RCRC) bit 5 ? Receive FIFO data request (RXDR) bit 4 ? Transmit FIFO data request (TXDR) bit 3 ? Data transfer over (DTO) bit 2 ? Command done (CD) bit 1 ? Response error (RE) bit 0 ? Card detect (CD)	0x0

### 16.5.24 RINTSTS

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
sdio_interrupt	[31:16]	R/W	Interrupt from SDIO card; one bit for each card. Bit[31] corresponds to Card[15], and bit[16] is for Card[0]. Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 0 ? No SDIO interrupt from card 1 ? SDIO interrupt from card Bits are logged regardless of interrupt-mask status.	0x0
int_status	[15:0]	R/W	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. bit 15 ? End-bit error (read)/write no CRC (EBE) bit 14 ? Auto command done (ACD) bit 13 ? Start-bit error (SBE) /Busy Clear Interrupt (BCI) bit 12 ? Hardware locked write error (HLE) bit 11 ? FIFO underrun/overrun error (FRUN) bit 10 ? Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9 ? Data read timeout (DRTO)/Boot Data Start (BDS) bit 8 ? Response timeout (RTO)/Boot Ack Received (BAR) bit 7 ? Data CRC error (DCRC) bit 6 ? Response CRC error (RCRC) bit 5 ? Receive FIFO data request (RXDR) bit 4 ? Transmit FIFO data request (TXDR) bit 3 ? Data transfer over (DTO) bit 2 ? Command done (CD) bit 1 ? Response error (RE) bit 0 ? Card detect (CD)	0x0

### 16.5.25 STATUS

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x48 Reset Value = 0x0000\_0406

Name	Bit	Type	Description	Reset Value
dma_req	[31]	R	DMA request signal state; either dw_dma_req or ge_dma_req, depending on DW-DMA or Generic-DMA selection.	0x0
dma_ack	[30]	R	DMA acknowledge signal state; either dw_dma_ack or ge_dma_ack, depending on DW-DMA or Generic-DMA selection.	0x0
fifo_count	[29:17]	R	FIFO count ? Number of filled locations in FIFO	0x0
response_index	[16:11]	R	Index of previous response, including any auto-stop sent by core	0x0
data_state_mc_busy	[10]	R	Data transmit or receive state-machine is busy	0x1
data_busy	[9]	R	Inverted version of raw selected card_data[0] 0 ? card data not busy 1 ? card data busy	0x0
data_3_status	[8]	R	Raw selected card_data[3]; checks whether card is present 0 ? card not present 1 ? card present	0x0
command fsm states	[7:4]	R	Command FSM states: 0 ? Idle 1 ? Send init sequence 2 ? Tx cmd start bit 3 ? Tx cmd tx bit 4 ? Tx cmd index + arg 5 ? Tx cmd crc7 6 ? Tx cmd end bit 7 ? Rx resp start bit 8 ? Rx resp IRQ response 9 ? Rx resp tx bit 10 ? Rx resp cmd idx 11 ? Rx resp data 12 ? Rx resp crc7 13 ? Rx resp end bit 14 ? Cmd path wait NCC 15 ? Wait; CMD-to-response turnaround NOTE: The command FSM state is represented using 19 bits. The STATUS Register(7:4) has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the STATUS(7:4) register. The three states that are not represented in the STATUS Register(7:4) are: * Bit 16 ? Wait for CCS * Bit 17 ? Send CCSD * Bit 18 ? Boot Mode Due to this, while command FSM is in "Wait for CCS" state or "Send CCSD" or "Boot Mode", the Status register indicates status as 0 for the bit field 7:4.	0x0
fifo_full	[3]	R	FIFO is full status	0x0
fifo_empty	[2]	R	FIFO is empty status	0x1
fifo_tx_watermark	[1]	R	FIFO reached Transmit watermark level; not qualified with data transfer.	0x1
fifo_rx_watermark	[0]	R	FIFO reached Receive watermark level; not qualified with data transfer.	0x0

### 16.5.26 FIFO TH

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x4C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	Reserved	-
DW_DMA_Mutiple_Transaction_Size	[30:28]	R/W	<p>Burst size of multiple transaction; should be programmed same as DW-DMA controller multiple-transaction-size SRC/DEST_MSIZE.</p> <p>000 ? 1 transfers 001 ? 4 010 ? 8 011 ? 16 100 ? 32 101 ? 64 110 ? 128 111 ? 256</p> <p>The units for transfers is the H_DATA_WIDTH parameter. A single transfer (dw_dma_single assertion in case of Non DW DMA interface) would be signalled based on this value.</p> <p>Value should be sub-multiple of <math>(RX\_WMark + 1) * (F\_DATA\_WIDTH/H\_DATA\_WIDTH)</math> and <math>(FIFO\_DEPTH - TX\_WMark) * (F\_DATA\_WIDTH/H\_DATA\_WIDTH)</math></p> <p>For example, if FIFO_DEPTH = 16, FDATA_WIDTH == H_DATA_WIDTH</p> <p>Allowed combinations for MSize and TX_WMark are:</p> <p>MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMark = 8 MSize = 4, TX_WMark = 4 MSize = 4, TX_WMark = 12 MSize = 8, TX_WMark = 8 MSize = 8, TX_WMark = 4</p> <p>Allowed combinations for MSize and RX_WMark are:</p> <p>MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMark = 3 MSize = 4, RX_WMark = 7 MSize = 4, RX_WMark = 11 MSize = 8, RX_WMark = 7</p> <p>Recommended: MSize = 8, TX_WMark = 8, RX_WMark = 7</p>	0x0
RX_WMark	[27:16]	R/W	<p>FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA-FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data.</p> <p>In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request.</p> <p>During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt.</p> <p>In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set.</p> <p>12 bits ? 1 bit less than FIFO-count of status register, which is 13 bits.</p> <p>Limitation: RX_WMark &lt;= FIFO_DEPTH-2</p> <p>Recommended: <math>(FIFO\_DEPTH/2) - 1</math>; (means greater than <math>(FIFO\_DEPTH/2) - 1</math>)</p> <p>NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS timeout.</p>	0x0
RSVD	[15:12]	-	Reserved	-

TX_WMark	[11:0]	R/W	<p>FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA-FIFO request is raised. If interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated, regardless of threshold programming.</p> <p>In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty).</p> <p>In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred.</p> <p>12 bits ? 1 bit less than FIFO-count of status register, which is 13 bits.</p> <p>Limitation: TX_WMark &gt;= 1;</p> <p>Recommended: FIFO_DEPTH/2; (means less than or equal to FIFO_DEPTH/2)</p>	0x0
----------	--------	-----	---	-----

### 16.5.27 CDETECT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:30]	-	Reserved	-
card_detect_n	[29:0]	R	Value on card_detect_n input ports (1 bit per card); read-only bits. 0 represents presence of card. Only NUM_CARDS number of bits are implemented.	0x0

### 16.5.28 WRTPRT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x54 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:30]	-	Reserved	-
write_protect	[29:0]	R	Value on card_write_prt input ports (1 bit per card). 1 represents write protection. Only NUM_CARDS number of bits are implemented.	0x0

### 16.5.29 GPIO

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x58 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
gpo	[23:8]	R/W	Value needed to be driven to gpo pins; this portion of register is read/write. Valid only when AREA_OPTIMIZED parameter is 0.	0x0
NSATT	[7:0]	R	Value on gpi input ports; this portion of register is read-only. Valid only when AREA_OPTIMIZED parameter is 0.	0x0

### 16.5.30 TCBCNT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x5C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
trans_card_byte_count	[31:0]	R	Number of bytes transferred by CIU unit to card. In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied. Both TCBCNT and TBBCNT share same coherency register. When AREA_OPTIMIZED parameter is 1, register should be read only after data transfer completes; during data transfer, register returns 0.	0x0

### 16.5.31 TBBCNT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x60 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
trans_fifo_byte_count	[31:0]	R	Number of bytes transferred between Host/DMA memory and BIU FIFO. In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied. Both TCBCNT and TBBCNT share same coherency register.	0x0

### 16.5.32 DEBNCE

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x64 Reset Value = 0x00FF\_FFFF

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
debounce_count	[23:0]	R/W	Number of host clocks (clk) used by debounce filter logic; typical debounce time is 5-25 ms.	0xFFFFFFF

### 16.5.33 USRID

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x68 Reset Value = 0x00DC\_2900

Name	Bit	Type	Description	Reset Value
USRID	[31:0]	R/W	User identification register; value set by user. Default reset value can be picked by user while configuring core before synthesis. Can also be used as scratch pad register by user.	0x00DC2900

### 16.5.34 VERID

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x6C Reset Value = 0x5342\_270A

Name	Bit	Type	Description	Reset Value
VERID	[31:0]	R	Synopsys version identification register; register value is hard-wired. Can be read by firmware to support different versions of core.	0x5342270A

### 16.5.35 HCON

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x70 Reset Value = Undefined

Name	Bit	Type	Description	Reset Value
HCON	[31:0]	R	<p>Hardware configurations selected by user before synthesizing core. Register values can be used to develop configuration-independent software drivers.</p> <p>bit 0 ? CARD_TYPE      0 ? MMC_ONLY      1 ? SD_MMC      bit [5:1] ? NUM_CARDS - 1      bit 6 ? H_BUS_TYPE      0 ? APB      1 ? AHB      bit [9:7] H_DATA_WIDTH      000 ? 16 bits      001 ? 32 bits      010 ? 64 bits      others ? reserved      bit [15:10] ? H_ADDR_WIDTH      0 to 7 ? reserved      8 ? 9 bits      9 ? 10 bits      Åæ      31 ? 32 bits      32 to 63 ? reserved+C132      bit[17:16] ? DMA_INTERFACE      00 ? none      01 ? DW_DMA      10 ? GENERIC_DMA      11 ? NON-DW-DMA      bit [20:18] GE_DMA_DATA_WIDTH      000 ? 16 bits      001 ? 32 bits      010 ? 64 bits      others ? reserved      bit 21 FIFO_RAM_INSIDE      0 ? outside      1 ? inside      bit 22 ? IMPLEMENT_HOLD_REG      0 ? no hold register      1 ? hold register      bit 23 ? SET_CLK_FALSE_PATH      0 ? no false path      1 ? false path set      bit [25:24] ? NUM_CLK_DIVIDER-1      bit 26 - AREA_OPTIMIZED      0 ? no area optimization      1 ? Area optimization      For 64-bit Address Configuration only      bit 27 - ADDR_CONFIG      0 ? 32-bit addressing supported      1 ? 64-bit addressing supported      For 32-bit Address Configuration only: bit [31:27] ? reserved (0)      For 64-bit Address Configuration only: bit [31:28] ? reserved (0)      For FIFO_DEPTH parameter, power-on value of RX_WMark value of FIFO Threshold Watermark Register represents FIFO_DEPTH - 1.</p>	:no reset

### 16.5.36 UHS\_REG

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x74 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DDR_REG	[31:16]	R/W	<p>DDR mode. These bits indicate DDR mode of operation to the core for the data transfer.</p> <p>0 ? Non-DDR mode 1 ? DDR mode</p> <p>UHS_REG [16] should be set for card number 0, UHS_REG [17] for card number 1 and so on.</p> <p>Note: Clear these bits in HS400 mode.</p>	0x0
VOLT_REG	[15:0]	R/W	<p>High Voltage mode. Determines the voltage fed to the buffers by an external voltage regulator.</p> <p>0 ? Buffers supplied with 3.3V Vdd 1 ? Buffers supplied with 1.8V Vdd</p> <p>These bits function as the output of the host controller and are fed to an external voltage regulator. The voltage regulator must switch the voltage of the buffers of a particular card to either 3.3V or 1.8V, depending on the value programmed in the register.</p> <p>VOLT_REG[0] should be set to 1'b1 for card number 0 in order to make it operate for 1.8V.</p>	0x0

### 16.5.37 RST\_n

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x78 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
CARD_RESET	[15:0]	R/W	<p>Hardware reset.</p> <p>1 ? Active mode 0 ? Reset</p> <p>These bits cause the cards to enter pre-idle state, which requires them to be re-initialized.</p> <p>- CARD_RESET[0] should be set to 1'b0 to reset card number 0</p> <p>- CARD_RESET[15] should be set to 1'b0 to reset card number 15.</p> <p>The number of bits implemented is restricted to NUM_CARDS.</p>	0x1

### 16.5.38 BMOD

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x80 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:11]	-	Reserved	-
PBL	[10:8]	R	<p>Programmable Burst Length. These bits indicate the maximum number of beats to be performed in one IDMAC transaction. The IDMAC will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. The permissible values are 1, 4, 8, 16, 32, 64, 128 and 256. This value is the mirror of MSIZE of FIFO TH register. In order to change this value, write the required value to FIFO TH register. This is an encode value as follows.</p> <p>000 ? 1 transfers      001 ? 4 transfers      010 ? 8 transfers      011 ? 16 transfers      100 ? 32 transfers      101 ? 64 transfers      110 ? 128 transfers      111 ? 256 transfers</p> <p>Transfer unit is either 16, 32, or 64 bits, based on HDATA_WIDTH. PBL is a read-only value and is applicable only for Data Access; it does not apply to descriptor accesses.</p>	0x0
DE	[7]	R/W	IDMAC Enable. When set, the IDMAC is enabled. DE is read/write.	0x0
DSL	[6:2]	R/W	Descriptor Skip Length. Specifies the number of Word/Word/D-word (depending on 16/32/64-bit bus) to skip between two un-chained descriptors. This is applicable only for dual buffer structure. DSL is read/write.	0x0
FB	[1]	R/W	Fixed Burst. Controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations. FB is read/write.	0x0
SWR	[0]	R/W	Software Reset. When set, the DMA Controller resets all its internal registers. SWR is read/write. It is automatically cleared after 1 clock cycle.	0x0

### 16.5.39 PLDMND

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x84 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PD	[31:0]	W	Poll Demand. If the OWN bit of a descriptor is not set, the FSM goes to the Suspend state. The host needs to write any value into this register for the IDMAC FSM to resume normal descriptor fetch operation. This is a write only register. PD bit is write-only.	0x0

### 16.5.40 DBADDRL

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x88 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SDLL	[31:0]	R/W	Start of Descriptor List Lower. Contains the LSB 32-bits of the base address of the First Descriptor.	0x0

### 16.5.41 DBADDRU

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x8C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SDLU	[31:0]	R/W	Start of Descriptor List Upper. Contains the MSB 32-bits of the base address of the First Descriptor.	0x0

### 16.5.42 IDSTS

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x90 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:17]	-	Reserved	-
FSM	[16:13]	R	DMAC FSM present state. 0 ? DMA_IDLE 1 ? DMA_SUSPEND 2 ? DESC_RD 3 ? DESC_CHK 4 ? DMA_RD_REQ_WAIT 5 ? DMA_WR_REQ_WAIT 6 ? DMA_RD 7 ? DMA_WR 8 ? DESC_CLOSE	0x0
FBE_CODE	[12:10]	R	Fatal Bus Error Code. Indicates the type of error that caused a Bus Error. Valid only with Fatal Bus Error bit?IDSTS[2] (IDSTS64[2], in case of 64-bit address configuration) set. This field does not generate an interrupt. 3Â»b001 ? Host Abort received during transmission 3Â»b010 ? Host Abort received during reception Others: Reserved	0x0
AIS	[9]	R/W	Abnormal Interrupt Summary. Logical OR of the following: - IDSTS[2] (IDSTS64[2], in case of 64-bit address configuration) ? Fatal Bus Interrupt - IDSTS[4] (IDSTS64[4], in case of 64-bit address configuration) ?DU bit Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. Writing a 1 clears this bit.	0x0
NIS	[8]	R/W	Normal Interrupt Summary. Logical OR of the following: - IDSTS[0] (IDSTS64[0], in case of 64-bit address configuration) ? Transmit Interrupt - IDSTS[1] (IDSTS64[1], in case of 64-bit address configuration) ? Receive Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes NIS to be set is cleared. Writing a 1 clears this bit.	0x0
RSVD	[7:6]	-	Reserved	-
CES	[5]	R/W	Card Error Summary. Indicates the status of the transaction to/from the card; also present in RINTSTS. Indicates the logical OR of the following bits: - EBE ? End Bit Error - RTO ? Response Timeout/Boot Ack Timeout - RCRC ? Response CRC - SBE ? Start Bit Error - DRTO ? Data Read Timeout/BDS timeout - DCRC ? Data CRC for Receive - RE ? Response Error Writing a 1 clears this bit. The abort condition of the IDMAC depends on the setting of this CES bit. If the CES bit is enabled, then the IDMAC aborts on a Â»response errorÂ»; however, it will not abort if the CES bit is cleared.	0x0
DU	[4]	R/W	Descriptor Unavailable Interrupt. This bit is set when the descriptor is unavailable due to OWN bit = 0 (DESO[31] =0). Writing a 1 clears this bit.	0x0
RSVD	[3]	-	Reserved	-
FBE	[2]	R/W	Fatal Bus Error Interrupt. Indicates that a Bus Error occurred (IDSTS[12:10]) (IDSTS64[12:10], in case of 64-bit address configuration). When this bit is set, the DMA disables all its bus accesses. Writing a 1 clears this bit.	0x0
RI	[1]	R/W	Receive Interrupt. Indicates the completion of data reception for a descriptor. Writing a 1 clears this bit.	0x0
TI	[0]	R/W	Transmit Interrupt. Indicates that data transmission is finished for a descriptor. Writing a 1 clears this bit.	0x0

### 16.5.43 IDINTEN

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x94 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:10]	-	Reserved	-
AI	[9]	R/W	<p>Abnormal Interrupt Summary Enable. When set, an abnormal interrupt is enabled.</p> <p>This bit enables the following bits:</p> <ul style="list-style-type: none"> <li>- IDINTEN[2] (IDINTEN64[2], in case of 64-bit address configuration) ? Fatal Bus Error Interrupt</li> <li>- IDINTEN[4] (IDINTEN64[4], in case of 64-bit address configuration) ? DU Interrupt</li> </ul>	0x0
NI	[8]	R/W	<p>Normal Interrupt Summary Enable. When set, a normal interrupt is enabled. When reset, a normal interrupt is disabled. This bit enables the following bits:</p> <ul style="list-style-type: none"> <li>- IDINTEN[0] (IDINTEN64[0], in case of 64-bit address configuration) ? Transmit Interrupt</li> <li>- IDINTEN[1] (IDINTEN64[1], in case of 64-bit address configuration) ? Receive Interrupt</li> </ul>	0x0
RSVD	[7:6]	-	Reserved	-
CES	[5]	R/W	Card Error summary Interrupt Enable. When set, it enables the Card Interrupt summary.	0x0
DU	[4]	R/W	Descriptor Unavailable Interrupt. When set along with Abnormal Interrupt Summary Enable, the DU interrupt is enabled.	0x0
RSVD	[3]	-	Reserved	-
FBE	[2]	R/W	Fatal Bus Error Enable. When set, the Fatal Bus Error Interrupt is enabled. When reset, Fatal Bus Error Enable Interrupt is disabled.	0x0
RI	[1]	R/W	Receive Interrupt Enable. When set, Receive Interrupt is enabled. When reset, Receive Interrupt is disabled.	0x0
TI	[0]	R/W	Transmit Interrupt Enable. When set, Transmit Interrupt is enabled. When reset, Transmit Interrupt is disabled.	0x0

### 16.5.44 DSCADDR

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x98 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
HDAL	[31:0]	R	Host Descriptor Address Pointer Lower 32-bits. Cleared on reset. Pointer updated by IDMAC during operation. This register points to the LSB 32-bits start address of the current descriptor read by the IDMAC.	0x0

### 16.5.45 DSCADDRU

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x9C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
HDAU	[31:0]	R	Host Descriptor Address Pointer Upper 32-bits. Cleared on reset. Pointer updated by IDMAC during operation. This register points to the MSB 32-bits start address of the current descriptor read by the IDMAC.	0x0

### 16.5.46 BUFADDRL

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0xA0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
HBAL	[31:0]	R	Host Buffer Address Pointer Upper 32 bits. Cleared on Reset. Pointer updated by IDMAC during operation. This register points to the current Data Buffer Address, upper 32 bits, being accessed by the IDMAC.	0x0

### 16.5.47 BUFADDRU

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0xA4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
HBAU	[31:0]	R	Host Buffer Address Pointer Upper 32 bits. Cleared on Reset. Pointer updated by IDMAC during operation. This register points to the current Data Buffer Address, upper 32 bits, being accessed by the IDMAC.	0x0

### 16.5.48 CardThrCtl

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x100 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:28]	-	Reserved	-
CardThreshold	[27:16]	R/W	Card Threshold size; N depends on the FIFO size: ÁaÁa N = 27, FIFO Size = 512 ÁaÁa N = 26, FIFO Size = 256 ÁaÁa N = 25, FIFO Size = 128 ÁaÁa N = 24, FIFO Size = 64 ÁaÁa N = 23, FIFO Size = 32 ÁaÁa N = 22, FIFO Size = 16 ÁaÁa N = 21, FIFO Size = 8 Note: This register is applicable when CardWrThrEn is set to '1' or CardRdThrEn is set to '1'	0x0
RSVD	[15:3]	-	Reserved	-
CardWrThrEn	[2]	R/W	Card Write Threshold Enable. Applicable when HS400 mode is enabled. 0 - Card write Threshold disabled 1 - Card write Threshold enabled Host Controller initiates Write Transfer only if Card Threshold amount of data is available in Transmit FIFO. Note: The write threshold is indicated by CardThreshold field.	0x0
BsyClrlntEn	[1]	R/W	Busy Clear Interrupt generation: 1'b0 - Busy Clear Interrupt disabled 1'b1 - Busy Clear Interrupt enabled Note: The application can disable this feature if it does not want to wait for a Busy Clear Interrupt. For example, in a multi-card scenario, the application can switch to the other card without waiting for a busy to be completed. In such cases, the application can use the polling method to determine the status of busy. By default this feature is disabled and backward-compatible to the legacy drivers where polling is used.	0x0
CardRdThrEn	[0]	R/W	Card Read Threshold Enable 1'b0 - Card Read Threshold disabled 1'b1 - Card Read Threshold enabled. Host Controller initiates Read Transfer only if Card Threshold amount of space is available in receive FIFO.	0x0

### 16.5.49 Back\_end\_power

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x104 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
Back End Power	[15:0]	R/W	Back end power 1'b0 ? Off; Reset 1'b1 ? Back-end Power supplied to card application; one pin per card	0x0

### 16.5.50 EMMC\_DDR\_REG

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x10C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
hs400_mode	[31]	R/W	HS400 Mode enable 1 - Enable 0 - Disable Note: The application is required to set this bit to '1' before initiating any data transfer CMD in HS400 mode. This bit shall be cleared by the host on exiting HS400 mode. In non HS400 mode, this bit shall be set to '0'.	0x0
RSVD	[30:16]	-	Reserved	-
HALF_START_BIT	[15:0]	R/W	Control for start bit detection mechanism inside DWC_mobile_storage based on duration of start bit; each bit refers to one slot. For eMMC 4.5, start bit can be: - Full cycle (HALF_START_BIT = 0) - Less than one full cycle (HALF_START_BIT = 1) Set HALF_START_BIT=1 for eMMC 4.5 and above; set to 0 for SD applications. Note: This bit is not applicable for HS400 mode. This bit is also ignored for a DDR operation when either of the parameter START_BIT_SAMPLING_DDR or ASYNC_SAMPLE_DATA_PATH is set to 1.	0x0

### 16.5.51 ENABLE\_SHIFT

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x110 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Enable_shift	[31:0]	R/W	Control for the amount of phase shift provided on the default enables in the design. Two bits are assigned for each card-slot. For example, bits[1:0] control slot0 and indicate the following. 00 ? Default phase shift 01 ? Enables shifted to next immediate positive edge 10 ? Enables shifted to next immediate negative edge 11 ? Reserved	0x0

### 16.5.52 TIEOFF\_MODE

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x400 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-

mode_hs400	[2]	R/W	HS400 Mode enable 1 - Enable 0 - Disable	0x0
mode_8bit	[1]	R/W	One bit per card indicates if card is 8-bit: 0 ? Non 8-bit mode 1 ? 8-bit mode	0x0
mode_4bit	[0]	R/W	One bit per card indicates if card is 1-bit or 4-bit: 0 ? 1-bit mode 1 ? 4-bit mode	0x0

### 16.5.53 TIEOFF\_SRAM

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x404 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
sram_sleep_n	[0]	R/W	Mobile Storage Host fifo sram sleep mode 0 - Sram sleep 1 - Sram awake	0x0

### 16.5.54 TIEOFF\_DRV\_PHASE

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x408 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
drv_shift	[2:0]	R/W	Selects drive clock among eight clocks 000 = 0 degree phase shifted clock 001 = 45 degree phase shifted clock 010 = 90 degree phase shifted clock 011 = 135 degree phase shifted clock 100 = 180 degree phase shifted clock 101 = 225 degree phase shifted clock 110 = 270 degree phase shifted clock 111 = 315 degree phase shifted clock	0x0

### 16.5.55 TIEOFF\_SMP\_PHASE

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x40C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
smp_shift	[2:0]	R/W	Selects sample clock among eight clocks 000 = 0 degree phase shifted clock 001 = 45 degree phase shifted clock 010 = 90 degree phase shifted clock 011 = 135 degree phase shifted clock 100 = 180 degree phase shifted clock 101 = 225 degree phase shifted clock 110 = 270 degree phase shifted clock 111 = 315 degree phase shifted clock	0x0

### 16.5.56 TIEOFF\_DS\_DELAY

- Base Address: 0x20930000(OSDMMC0)
- Base Address: 0x20940000(OSDMMC1)
- Address = Base Address + 0x410 Reset Value = 0x0000\_0040

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
ds_shift	[8:0]	R/W	This field is used for data strobe clock fine grain adjustment by setting the number of steps (single step is approximately 200 ? 500 ps)	0x40

# 17 DMA

## 17.1 Overview

DRONE\_SOC supports five Direct Memory Access (DMA) tops:

- Two Peripheral-to-Memory transfer and vice-versa (DMA0 and DMA1)

## 17.2 Features

- The key features of DMA Controller are :
- Table 17.1 lists the key features of DMA controllers.

Features	DMA0	DMA1
Supports Data size	Up to word (32-bit)	Up to word (32-bit)
Supports Burst size	Up to 16 burst	Up to 16 burst
Supports channel	8 channels At the same time	8 channels At the same time

**Table 17.1:** Key Features of the DMA Controllers

- Refer these features for DMA and for writing DMA assembly code.

**NOTE:** DMA Controller sends only one interrupt to Interrupt Controller for each DMA although each DMA module has 32 interrupt sources.

- Table 17.2 and Table 17.3 describes DMA request mapping table.

## 17.3 Peripheral DMA Request ID

### 17.3.1 DMAC0

Index	Description	Index	Description
0	SPI0.TX	16	CRYPTO.HR
1	SPI0.RX	17	-
2	SPI2.TX	18	-
3	SPI2.RX	19	-
4	QSPI1.TX	20	-
5	QSPI1.RX	21	-
6	QSPI3.TX	22	-
7	QSPI3.RX	23	-
8	UART0.TX	24	-
9	UART0.RX	25	-
10	USART0.TX	26	-
11	USART0.RX	27	-
12	USART2.TX	28	-
13	USART2.RX	29	-
14	CRYPTO.BR	30	-
15	CRYPTO.BW	31	-

**Table 17.2:** Peripheral DMAC0 Request ID

### 17.3.2 DMAC1

Index	Description	Index	Description
0	SPI1.TXI	16	-
1	SPI1.RXI	17	-
2	QSPI0.TX	18	-
3	QSPI0.RX	19	-
4	QSPI2.TX	20	-
5	QSPI2.RX	21	-
6	QSPI4.TX	22	-
7	QSPI4.RX	23	-
8	UART1.TX	24	-
9	UART1.RX	25	-
10	USART1.TX	26	-
11	USART1.RX	27	-
12	USART3.TX	28	-
13	USART3.RX	29	-
14		30	-
15		31	-

**Table 17.3:** Peripheral DMAC1 Request ID

## 17.4 Instruction

Table 17.4 summarizes the instruction syntax.

Mnemonic	Instruction	Thread Usage: M = DMA Manager C = DMA Channel		Description
DMAADDH	Add Half-word	-	C	Adds an immediate 16-bit value to the SARn Register or DARn Register for the DMA channel thread. This value enables the DMAC to support 2D DMA Operations. Refer to Source Address Registers (SARn) and Destination Address Registers (DARn) for more information.
DMAEND	End	M	C	Signals the DMAC that the DMA sequence is complete. After completing all DMA transfers for the DMA channel, DMAC moves the channel to the Stopped state. It also flushes data from MFIFO and invalidates all cache entries for the thread.
DMAFLUSHP	Flush and notify Peripheral	-	C	Clears the state in DMAC that describes the contents of the peripheral and sends a message to the peripheral to resend its level status.
DMAGO	Go	M	-	Performs these steps on the DMA channel when the DMA manager executes this instruction for a DMA channel that is in the Stopped state: 1. Moves a 32-bit immediate into the program counter. 2. Sets security state of DMA channel. 3. Updates DMA channel to the executing state.
DMALD	Load	-	C	Instructs DMAC to perform a DMA load by using AXI transactions that SARn and CCRn specify. It sends the Read data to MFIFO and tags it with the corresponding channel number. DMALD is an unconditional instruction. However, DMALDS and DMALDB are conditional on the state of the request_type flag. If the src_inc bit in the Channel Control Registers (refer to CCRn for more information) is set to incrementing, the DMAC updates SARn after executing DMALD.

Mnemonic	Instruction	Thread Usage: M = DMA Manager C = DMA Channel	Description	
DMALDP	Load Peripheral	-	C	<p>Instructs DMAC to perform a DMA load by using AXI transactions that SARn and CCRn specify. It sends the Read data to a FIFO that is tagged with the corresponding channel number. After it receives the last data item, it updates data type[1:0] to signal the peripheral that the data transfer is complete.</p> <p>If the src_inc bit in the CCR is set to incrementing, the DMAC updates SARn after DMAC executes DMALDP.</p>
DMALP	Loop	-	C	<p>Instructs DMAC to load an 8-bit value into the Loop Counter Register that you specify. This instruction indicates the start of an instruction set.</p> <p>Set the end of the instructions set by using the DMALPEND instruction.</p> <p>Refer to DMALPEND[S B] for more information. DMAC repeats the set of instructions, which you insert between DMALP and DMALPEND, until the value in the Loop Counter Register reaches 0.</p>
DMALPEND	Loop End	-	C	<p>Indicates the last instruction in the program loop.</p> <p>However, the behavior of the DMAC depends on whether DMALP or DMALPFE starts the loop. If a loop starts with DMALP or DMALPFE, then:</p> <ul style="list-style-type: none"> <li>* DMALP: The loop has a defined loop count and DMALPEND[S B] instructs the DMAC to read the value of the Loop Counter register. If a Loop Counter register returns:           <ul style="list-style-type: none"> <li>- Zero: DMAC executes a DMANOP and exits the loop.</li> <li>- Non-zero: DMAC decrements the value in the Loop Counter register and updates the thread PC to contain the address of the first instruction in the program loop, that is, the instruction that follows the DMALP.</li> </ul> </li> <li>DMALPFE: The loop has an undefined loop count and the DMAC uses the state of the request_last flag to control when it exits the loop.</li> </ul> <p>If the request_last flag is:</p> <ul style="list-style-type: none"> <li>0 =DMAC updates the thread PC to contain the address of the first instruction in the program loop, that is, the instruction that follows the DMALP.</li> <li>1 =DMAC executes a DMANOP and therefore exits the Loop</li> </ul>

Mnemonic	Instruction	Thread Usage: M = DMA Manager C = DMA Channel		Description
DMALPFE	Loop Forever	-	C	The assembler uses Loop Forever to configure certain bits in DMALPEND. Refer to DMALPEND[S B] for more information.
DMAKILL	Kill	M	C	<p>Instructs DMAC to terminate the execution of a thread immediately. DMAC performs these steps depending on the type of thread:</p> <p>DMA manager thread:</p> <ol style="list-style-type: none"> <li>1. Invalidates all cache entries for the DMA manager.</li> <li>2. Moves the DMA manager to the Stopped state.</li> </ol> <p>DMA channel thread:</p> <ol style="list-style-type: none"> <li>1. Moves the DMA channel to the Killing state.</li> <li>2. Waits for AXI transactions, with an ID equal to the DMA channel number to be completed.</li> <li>3. Invalidates all cache entries for the DMA channel.</li> <li>4. Removes all entries in MFIFO for the DMA channel.</li> <li>5. Removes all entries in the Read buffer queue and Write buffer queue for the DMA channel.</li> <li>6. Moves the DMA channel to the Stopped state.</li> </ol>
DMAMOV	Move	-	C	Instructs DMAC to move a 32-bit immediately into these Registers: SARn, DARn, and CCRn.
DMANOP	No Operation	M	C	Use this instruction for code alignment because it does not have specific instructions.
DMARMB	Read Memory Barrier	-	C	Forces the DMA channel to wait until all the executed DMA LD instructions for that channel have been issued and completed on the AXI master interface. This enables Write-after-Read sequences to the same address location with no hazards.
DMASEV	Send Event	M	C	<p>Instructs DMAC to modify an event-interrupt resource. Depending on how you program the Interrupt Enable Register (INTEN), this instruction either:</p> <ul style="list-style-type: none"> <li>* Generates event &lt;event_num&gt;</li> <li>or</li> <li>* Signals an interrupt by using irq&lt;event_num&gt;</li> </ul>

Mnemonic	Instruction	Thread Usage: M = DMA Manager C = DMA Channel	Description	
DMAST	Store	-	C	Instructs DMAC to transfer data from FIFO to the location that DARn specifies by using AXI transactions that the DA Register and CCRn specify. When the dst_inc bit in CCRn is set to incrementing, DMAC updates DAR after it executes DMAST[S B].
DMASTP	Store and notify Peripheral	-	C	Instructs DMAC to transfer data from FIFO to the location that the DARn specifies by using AXI transactions that the DA Register and CCRn specify. DMASTP uses the DMA channel number to access the appropriate location in FIFO. After the DMA store is complete, and DMAC receives a buffered Write response, DMASTP updates datatype[1:0] to notify the peripheral that the data transfer is complete. When the dst_inc bit in CCRn is set to incrementing, DMAC updates DAR on after it executes DMASTP<S B>.
DMASTZ	Store Zero	-	C	Instructs DMAC to store zeros by using AXI transactions that DARn and CCRn specify. When the dst_inc bit in the CCRn is set to incrementing, DMAC updates the DARn after it executes DMASTZ.
DMAWFE	Wait for Event	M	C	Instructs DMAC to halt execution of the thread until event_num specifies the event to occur. When the event occurs, the thread moves to the executing state and DMAC clears the event.
DMAWFP	Wait for Peripheral	-	C	Instructs DMAC to halt execution of the thread until the specified peripheral signals a DMA request for that DMA channel.
DMAWMB	Write Memory Barrier	-	C	Forces the DMA channel to wait until all the executed DMAST instructions for that channel have been issued and completed on the AXI master interface. This enables Read-after-Write sequences to the same address location without any hazards.

Table 17.4: Instruction Syntax Summary

**NOTE:** Each DMA330 comprises a manager thread and eight channel threads. The manager thread controls the overall operation of DMAC which includes initiating and killing the channel. The channel threads operate the DMA.

### 17.4.1 Key Instruction

To run the channel thread, you must write the assembly code. Table 17.4 describes the instructions.

#### 17.4.1.1 DMAMOV

Use the Move command to instruct the DMAC to move 32-bit width immediately into Source Address Register(SAR), Destination Address Register (DAR), and Channel Control Register (CCR).

- SAR
  - Example: DMAMOV SAR, 0x2400\_0000
  - 0x2400\_0000 is the source address of DMA operation
- DAR
  - Example: DMAMOV DAR, 0x2400\_1000
  - 0x2400\_1000 is destination address of DMA operation
- CCR
  - Example: DMAMOV CCR, SB2 SS32 SP0 DB2 DS32 DP0
  - Source: Burst length is 2 and 32-bit data width
  - Destination: Burst length is 2 and 32-bit data width
  - SP0 and DP0 indicate normal and secure, respectively. SP2 and DP2 indicate normal and non-secure, respectively.
  - Refer to Table 17.5 to for more information on the exact DMA settings, such as burst length, bit-width, address increment, and so on.
  - Table 17.5 describes DMAMOV CCR argument description and the default values.

Syntax	Description	Options	Default
SA123	Source address increment Sets the value of ARBURST[0].	I = Increments the source address F = Source address is fixed	I
SS	Source burst size in bits Sets the value of ARSIZE[2:0].	8, 16, 32, 64, or 128	8
SB	Source burst length Sets the value of ARLEN[3:0].	1 to 16	1
SP	Source protection	0 to 7	0
SC	Source cache	0 to 15	0
DA	Destination address increment Sets the value of AWBURST[0].	I = Increments the destination address F = Destination address is fixed	I
DS	Destination burst size in bits Sets the value of AWSIZE[2:0].	8, 16, 32, 64, or 128	8
DB	Destination burst length Sets the value of AWLEN[3:0].	1 to 16	1
DP	Destination protection	0 to 7	0
DC	Destination cache	0 to 15	0
ES	Endian swap size, in bits	8, 16, 32, 64, or 128	8

**Table 17.5:** DMAMOV CCR Argument Description and the Default Values**NOTE:**

1. You must use decimal values when programming this immediate value.
2. The assembler always sets bit 3 to 0 and uses bits[2:0] of your chosen value for SC because the DMAC ties ARCACHE[3] to Low. Refer to CCRn register bit assignments for more information.
3. The assembler always sets bit 2 to 0 and uses bit[3] and bits[1:0] of your chosen value for DC because the DMAC ties AWCACHE[2] to Low. Refer to CCRn register bit assignments for more information.

#### 17.4.1.2 DMAMOV, DMALDP

Use the Load command to instruct the DMAC to perform DMA load by using AXI transactions that SAR and CCR specifies. For example, if you define CCR as 32-bit bus and burst length as 2, DMALD generates a bus transaction of 32-bit and burst length 2. DMALDP notifies the peripheral when the data transfer is complete.

#### 17.4.1.3 DMAST, DMASTP

Use the Store command to instruct the DMAC to transfer data from FIFO to a location that DAR specifies by using AXI transactions. DAR and CCR specify the AXI transactions. For example, if you define CCR as 32-bit and burst length as 2, DMAST generates a 32-bit bus transaction and burst length 2. DMASTP notifies the peripheral when the data transfer is complete.

#### 17.4.1.4 DMASTZ

Use the Store Zero command to instruct the DMAC to store zeros by using AXI transactions that DAR and CCR specifies. For example, if you define CCR as 32-bit bus transaction and burst length as 2, DMASTZ generates 32-bit bus transaction and burst length 2 with zeros at data bus.

#### 17.4.1.5 DMALP, DMALPEND

DMALP lc0, 4[code] to DMALPEND lc0 loops (iterates) the [code] four times. The two loop counters are:

- lc0
- lc1

You can use nested loop by two loop counters.

#### 17.4.1.6 DMAWFP

This key instruction is used for peripheral DMA. Wait for Peripheral instructs DMAC to stop execution of the thread until the specified peripheral signals a DMA request for that DMA channel.

#### 17.4.1.7 DMAFLUSHP

This key instruction is used for peripheral DMA. Flush Peripheral which describes the contents of the peripheral, clears the state in DMA. It also sends a message to the peripheral to resend its level status. This instruction asserts DMAACK. When you require DMAACK at a certain point, place this instruction at that point.

#### 17.4.1.8 DMAEND

DMAEND instructs to stop a channel.

### 17.4.2 USAGE Model

DMA330 requires its own binary instruction code.

To perform a USAGE model:

1. Load DMA binary into memory.
2. Use DMA debug SFRs to start DMA controller, DMA330.

#### 17.4.2.1 Using debug SFRs

Write DBGCMD, DBGINST0, and DBGINST1 (all write-only). Before Writing these SFRs, verify if DBGSTATUS is busy or not. DBGINST0 and DBGINST1 include debug instructions. These SFRs receive only three instructions: DMAGO, DMASEV, and DMAKILL. DMAGO starts a channel. DBGCMD executes the instruction stored in the DBGINST0 and SFRs.

Example 17.1 shows the USAGE model.

```
Load channel control Register
Single transfer , 32-bit/non-secure
DMAMOV      CCR, SB1 SS32 SP2 DB1 DS32 DP2
              ; SB1, DB1          : Burst length: 1
              ; SS32, DS32        : 32-bit Data I/F
              ; SP0, DP0          : Secure access
              ; SP2, DP2          : Non-secure access

In case of Peripheral transfer , should initialize peripheral
DMAFLUSHP 0

Source: IntRAM0 , Destination: IntRAM1
DMAMOV      SAR, 0xd0020000
DMAMOV      DAR, 0xd0028000
DMALP       Ic0 , 32
DMA LD
DMA ST
DMA LPEND   Ic0
DMA SEV     E0
DMA END
```

Example 17.1: USAGE Model

#### 17.4.2.2 Security Scheme

MDMA and PDMA run in the non-secure mode only.

- Channel thread
  - MDMAC: Runs in non-secure (ns bit at DMAGO instruction is 1) mode only
  - DMAC : Runs in non-secure (ns bit at DMAGO instruction is 1) mode only
- ASM code
  - For non-secure transaction:
    - \* Use SP2 and DP2 at DMAMOV instruction
    - \* APROT[1] will be 1'b1

#### 17.4.2.3 Interrupts

DMAC provides IRQ signals for use as level sensitive interrupts to external CPUs. When you program the Interrupt Enable Register to generate an interrupt after DMAC executes DMASEV, it sets the corresponding IRQ as High.

You can clear the interrupt by writing to the Interrupt Clear Register.

To control the interrupt:

1. Set up the Interrupt Enable Register to generate interrupts:
  - The interrupt enable Register is a 32-bit Register. Each bit of the INTEN Registers verifies if DMAC signals an interrupt by using the corresponding IRQ.
  - Program the appropriate bit to control the DMAC response on executing DMASEV.
    - Bit[N] = 0: When executing DMASEV for event N, the DMAC signals event N to all the threads.
    - Bit[N] = 1: When executing DMASEV for event N, the DMAC sets irq[N] as High.
2. To set the corresponding IRQ to high by executing DMASEV, program the assembly code.
  - Use DMASEV instruction to interrupt one of the IRQ outputs.
3. Clear the interrupt by Writing to the Interrupt Clear Register.
  - Each bit in the INTCLR Register controls the clearing of an interrupt.
  - Program to control the clearing of the IRQ outputs:
    - Bit[N] = 0: The status of irq[N] does not change. Bit[N] = 1: The DMAC sets irq[N] as low.
    - When DMA is set to a fault status, an interrupt occurs.

#### 17.4.2.4 Summary

This section summarizes configuring DMAC and the channel thread.

- Configuring DMAC:  
You can configure up to eight DMA channels with the DMAC. Each channel can support a single concurrent thread of DMA operation. Additionally, there is a single DMA manager thread to initialize the DMA channel thread.

- Channel thread:

Each channel thread can operate DMA. Ensure to write the assembly code accordingly. When you require a number of independent DMA channels, you must write a number of assembly codes for each channel. Assemble the code, link the code on a file, and load this file into the memory.

## 17.5 Register Description

Most Special Function Registers (SFRs) are Read-only. The main function of SFR is to verify the DMA330 status. There are multiple SFRs for DMA330. This section describes DRONE\_SOCspecific SFRs only. Refer to Chapter 3 DMA330 TRM from ARM for more information.

### 17.5.1 Register Map Summary

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)

Register	Offset	Description	Reset Value
DSR	0x00 0x00	Specifies the DMA Status Register	0x0000_0200
DPC	0x04 0x04	Specifies the DMA Program Counter Register	0x0000_0000
INTEN	0x20 0x20	Specifies the Interrupt Enable Register	0x0000_0000
INT_EVENT_RIS	0x24 0x24	Specifies the Event Status Register	0x0000_0000
INTMIS	0x28 0x28	Specifies the Interrupt Status Register	0x0000_0000
INTCLR	0x2C 0x2C	Specifies the Interrupt Clear Register	0x0000_0000
FSRD	0x30 0x30	Specifies the Fault Status DMA Manager Register	0x0000_0000
FSRC	0x34 0x34	Specifies the Fault Status DMA Channel Register	0x0000_0000
FTRD	0x38 0x38	Specifies the fault type DMA Manager Register	0x0000_0000
FTR0	0x40 0x40	Specifies the Fault Type for DMA Channel 0	0x0000_0000
FTR2	0x44 0x44	Specifies the Fault Type for DMA Channel 2	0x0000_0000
FTR2	0x48 0x48	Specifies the Fault Type for DMA Channel 2	0x0000_0000
FTR3	0x4C 0x4C	Specifies the Fault Type for DMA Channel 3	0x0000_0000
FTR4	0x50 0x50	Not available	0x0000_0000
FTR5	0x54 0x54	Not available	0x0000_0000
FTR6	0x58 0x58	Not available	0x0000_0000
FTR7	0x5C 0x5C	Not available	0x0000_0000
CSR0	0x100 0x100	Specifies the Channel Status for DMA Channel 0	0x0000_0000
CPC0	0x104 0x104	Specifies the Channel PC for DMA Channel 0	0x0000_0000
CSR1	0x108 0x108	Specifies the Channel Status for DMA Channel 1	0x0000_0000
CPC1	0x10C 0x10C	Specifies the Channel PC for DMA Channel 1	0x0000_0000
CSR2	0x110 0x110	Specifies the Channel Status for DMA Channel 2	0x0000_0000
CPC2	0x114 0x114	Specifies the Channel PC for DMA Channel 2	0x0000_0000
CSR3	0x118 0x118	Specifies the Channel Status for DMA Channel 3	0x0000_0000

CPC3	0x11C 0x11C	Specifies the Channel PC for DMA Channel 3	0x0000_0000
CSR4	0x120 0x120	Not available	0x0000_0000
CPC4	0x124 0x124	Not available	0x0000_0000
CSR5	0x128 0x128	Not available	0x0000_0000
CPC5	0x12C 0x12C	Not available	0x0000_0000
CSR6	0x130 0x130	Not available	0x0000_0000
CPC6	0x134 0x134	Not available	0x0000_0000
CSR7	0x138 0x138	Not available	0x0000_0000
CPC7	0x13C 0x13C	Not available	0x0000_0000
SAR0	0x400 0x400	Specifies the Source Address for DMA Channel 0	0x0000_0000
DAR0	0x404 0x404	Specifies the Destination Address for DMA Channel 0	0x0000_0000
CCR0	0x408 0x408	Specifies the Channel Control for DMA Channel 0	0x0000_0000
LC0_0	0x40C 0x40C	Specifies the Loop Counter 0 for DMA Channel 0	0x0000_0000
LC1_0	0x410 0x410	Specifies the Loop Counter 1 for DMA Channel 0	0x0000_0000
SAR2	0x420 0x420	Specifies the Source Address for DMA Channel 2	0x0000_0000
DAR2	0x424 0x424	Specifies the Destination Address for DMA Channel 2	0x0000_0000
CCR2	0x428 0x428	Specifies the Channel Control for DMA Channel 2	0x0000_0000
LC0_2	0x42C 0x42C	Specifies the Loop Counter 0 for DMA Channel 2	0x0000_0000
LC1_2	0x430 0x430	Specifies the Loop Counter 1 for DMA Channel 2	0x0000_0000
SAR2	0x440 0x440	Specifies the Source Address for DMA Channel 2	0x0000_0000
DAR2	0x444 0x444	Specifies the Destination Address for DMA Channel 2	0x0000_0000
CCR2	0x448 0x448	Specifies the Channel Control for DMA Channel 2	0x0000_0000
LC0_2	0x44C 0x44C	Specifies the Loop Counter 0 for DMA Channel 2	0x0000_0000
LC1_2	0x450 0x450	Specifies the Loop Counter 1 for DMA Channel 2	0x0000_0000
SAR3	0x460 0x460	Specifies the Source Address for DMA Channel 3	0x0000_0000
DAR3	0x464 0x464	Specifies the Destination Address for DMA Channel 3	0x0000_0000
CCR3	0x468 0x468	Specifies the Channel Control for DMA Channel 3	0x0000_0000
LC0_3	0x46C 0x46C	Specifies the Loop Counter 0 for DMA Channel 3	0x0000_0000
LC1_3	0x470 0x470	Specifies the Loop Counter 1 for DMA Channel 3	0x0000_0000
SAR4	0x480 0x480	Not available	0x0000_0000
DAR4	0x484 0x484	Not available	0x0000_0000
CCR4	0x488 0x488	Not available	0x0000_0000
LC0_4	0x48C 0x48C	Not available	0x0000_0000

LC1_4	0x490 0x490	Not available	0x0000_0000
SAR5	0x4A0 0x4A0	Not available	0x0000_0000
DAR5	0x4A4 0x4A4	Not available	0x0000_0000
CCR5	0x4A8 0x4A8	Not available	0x0000_0000
LC0_5	0x4AC 0x4AC	Not available	0x0000_0000
LC1_5	0x4B0 0x4B0	Not available	0x0000_0000
SAR6	0x4C0 0x4C0	Not available	0x0000_0000
DAR6	0x4C4 0x4C4	Not available	0x0000_0000
CCR6	0x4C8 0x4C8	Not available	0x0000_0000
LC0_6	0x4CC 0x4CC	Not available	0x0000_0000
LC1_6	0x4D0 0x4D0	Not available	0x0000_0000
SAR7	0x4E0 0x4E0	Not available	0x0000_0000
DAR7	0x4E4 0x4E4	Not available	0x0000_0000
CCR7	0x4E8 0x4E8	Not available	0x0000_0000
LC0_7	0x4EC 0x4EC	Not available	0x0000_0000
LC1_7	0x4F0 0x4F0	Not available	0x0000_0000
DBGSTATUS	0xD00 0xD00	Specifies the Debug Status Register	0x0000_0000
DBGCMD	0xD04 0xD04	Specifies the Debug Command Register	0x0000_0000
DBGINST0	0xD08 0xD08	Specifies the Debug Instruction 0 Register	0x0000_0000
DBGINST1	0xD0C 0xD0C	Specifies the Debug Instruction 1 Register	0x0000_0000
CR0	0xE00 0xE00	Specifies the Configuration Register 0	0x003F_F035
CR1	0xE04 0xE04	Specifies the Configuration Register 1	0x0000_0074
CR2	0xE08 0xE08	Specifies the Configuration Register 2	0x0000_0000
CR3	0xE0C 0xE0C	Specifies the Configuration Register 3	0xFFFF_FFFF
CR4	0xE10 0xE10	Specifies the Configuration Register 4	0xFFFF_FFFF
CRD	0xE14 0xE14	Specifies the Configuration Register Dn	0x03F7_3732
WD	0xE80 0xE80	Watchdog Register	0x0000_0000
periph_id_0	0xFE0 0xFE0	Specifies the Peripheral Identification Registers 0	0x0000_0030
periph_id_2	0xFE4 0xFE4	Specifies the Peripheral Identification Registers 2	0x0000_0013
periph_id_2	0xFE8 0xFE8	Specifies the Peripheral Identification Registers 2	0x0000_0034
periph_id_3	0xFEC 0xFEC	Specifies the Peripheral Identification Registers 3	0x0000_0000
pcell_id_0	0xFF0 0xFF0	Specifies the PrimeCell Identification Registers 0	0x0000_000D
pcell_id_2	0xFF4 0xFF4	Specifies the PrimeCell Identification Registers 2	0x0000_00F0

pcell_id_2	0xFF8 0xFF8	Specifies the PrimeCell Identification Registers 2	0x0000_0005
pcell_id_3	0xFFC 0xFFC	Specifies the PrimeCell Identification Registers 3	0x0000_00B1

### 17.5.2 DSR

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x00, 0x00 Reset Value = 0x0000\_0200

Name	Bit	Type	Description	Reset Value
RSVD	[31:10]	-	Reserved	-
DNS	[9]	R	Provides the security status of the DMA manager thread 0 = DMA manager operates in secure state 1 = DMA manager operates in non-secure state *NOTE: Reset value of SEC_DMA is 0x0	0x1
Wakeup_event	[8:4]	R	When the DMA manager thread executes a DMAWFE instruction, it waits for these events to occur: b00000 = Event[0] b00001 = Event[1] b00010 = Event[2] : b11111 = Event[31]	0x0
DMA status	[3:0]	R	The operating state of the DMA manager b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = 'Waiting for event' b0101-b1110 = Reserved b1111 = Faulting	0x0

### 17.5.3 DPC

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x04, 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
pc_mgr	[31:0]	R	Program counter for the DMA manager thread	0x0

### 17.5.4 INTEN

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x20, 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
event_irq_select	[31:0]	R/W	<p>Programs the appropriate bit to control the DMAC Responses when it executes DMASEV</p> <p>*Bit[N] 0 = When the DMAC executes DMASEV for the event-interrupt resource N, DMAC signals event N to all of the threads. Set bit[N] to 0 if your system design does not use irq[N] to signal an interrupt request.</p> <p>*Bit[N] 1 = When DMAC executes DMASEV for the event-interrupt resource N, the DMAC sets irq[N] High Set bit[N] to 1 if your system design requires irq[N] to signal an interrupt request</p>	0x0

### 17.5.5 INT\_EVENT\_RIS

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x24, 0x24 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DMASEV active	[31:0]	R	<p>Returns the status of the event-interrupt resources</p> <p>*Bit[N] 0 = Event N is inactive or irq[N] is Low *Bit[N] 1 = Event N is active or irq[N] is High</p>	0x0

### 17.5.6 INTMIS

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x28, 0x28 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
irq_status	[31:0]	R	<p>Provides the status of the interrupts that are active in DMAC</p> <p>*Bit[N] 0 = Interrupt N is inactive and therefore irq[N] is Low *Bit[N] 1 = Interrupt N is active and therefore irq[N] is High</p> <p>NOTE: You must use the INTCLR Register to set bit[N] to 0. Refer to Interrupt Clear Register (INTCLR) for more information.</p>	0x0

### 17.5.7 INTCLR

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x2C, 0x2C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
irq_clr	[31:0]	W	Controls the clearing of the irq outputs *Bit[N] 0 = The status of irq[N] does not change *Bit[N] 1 = DMAC sets irq[N] to Low if the INTEN Register programs the DMAC to signal an interrupt Else, the status of irq[N] does not change.	0x0

### 17.5.8 FSRD

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x30, 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
fs_mgr	[0]	R	Provides the fault status of the DMA manager read as: 0 = The DMA manager thread is not in the faulting state 1 = The DMA manager thread is in the faulting state	0x0

### 17.5.9 FSRC

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x34, 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
fault_status	[7:0]	R	Each bit provides the fault status of the corresponding channel Read as: *Bit[N] 0 = No fault is present on DMA channel N *Bit[N] 1 = DMA channel N is in the faulting or faulting completing state	0x0

### 17.5.10 FTRD

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x38, 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	Reserved	-
dbg_instr	[30]	R	When the DMA manager aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface. 0 = Reads instruction that generates an abort from system memory 1 = Reads instruction that generates an abort from the debug interface	0x0
RSVD	[29:17]	-	Reserved	-
Instr_fetch_err	[16]	R	Indicates the AXI response that DMAC receives on the RRESP bus after the DMA manager performs a fetch instruction: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response	0x0
RSVD	[15:6]	-	Reserved	-
mgr_evnt_err	[5]	R	Indicates whether the DMA manager was attempting to execute DMAWFE or DMASEV with inappropriate security permissions 0 = The DMA manager has appropriate security to execute DMAWFE or DMASEV 1 = A DMA manager thread in the son-secure state attempts to execute either: * DMAWFE to wait for a secure event or * DMASEV to create a secure event or secure interrupt	0x0
dmago_err	[4]	R	Indicates if the DMA manager was attempting to execute DMAGO with inappropriate security permissions 0 = The DMA manager has appropriate security to execute DMAGO 1 = A DMA manager thread in the non-secure state attempts to execute DMAGO to create a DMA channel that operates in the secure state	0x0
RSVD	[3:2]	-	Reserved	-
operand_invalid	[1]	R	Indicates if the DMA manager attempts to execute an instruction operand that is invalid for the DMAC configuration 0 = Valid operand 1 = Invalid operand	0x0
undef_instr	[0]	R	Indicates if the DMA manager attempts to execute an undefined instruction 0 = Defined instruction 1 = Undefined instruction	0x0

### 17.5.11 FTR0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x40, 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
lockup_err	[31]	R	Indicates if the DMA channel has been locked up due to resource starvation 0 = The DMA channel has adequate resources 1 = The DMA channel has locked up because of inadequate resources This fault is an imprecise abort.	0x0

dbg_instr	[30]	R	<p>When the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface.</p> <p>0 = Reads instruction that generates an abort from system memory 1 = Reads instruction that generates an abort from the debug interface</p> <p>This fault is an imprecise abort. However, the bit is only valid when a precise abort occurs.</p>	0x0
RSVD	[29:19]	-	Reserved	-
data_read_err	[18]	R	<p>Indicates the AXI response that the DMAC receives on the RRESP bus after the DMA channel thread performs a data Read</p> <p>0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response.</p> <p>This fault is an imprecise abort.</p>	0x0
data_write_err	[17]	R	<p>Indicates the AXI response that the DMAC receives on the BRESP bus after the DMA channel thread performs a data Write</p> <p>0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response.</p> <p>This fault is an imprecise abort.</p>	0x0
instr_fetch_err	[16]	R	<p>Indicates the AXI response that the DMAC receives on the RRESP bus after the DMA channel thread performs an instruction fetch</p> <p>0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response.</p> <p>This fault is a precise abort.</p>	0x0
RSVD	[15:14]	-	Reserved	-
st_data_unavailable	[13]	R	<p>Indicates if the MFIFO does not contain the data to enable the DMAC to perform the DMAST</p> <p>0 = Indicates that MFIFO contains all the data to enable completion of DMAST 1 = Indicates that previous DMALDs have not placed enough data in the MFIFO to enable completion of DMAST</p> <p>This fault is a precise abort.</p>	0x0
mfifo_err	[12]	R	<p>Indicates if MFIFO prevents the DMA channel thread from executing DMALD or DMAST Depending on the instruction: *DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is very small to hold the data that DMALD requires *DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is very small to store the data to enable DMAST to complete.</p> <p>This fault is an imprecise abort.</p>	0x0
RSVD	[11:8]	-	Reserved	-
ch_rdwr_err	[7]	R	<p>Indicates if a DMA channel thread, in the nonsecure state, attempts to program CCRn Register to perform a secure read or secure write</p> <p>0 = A DMA channel thread in the non-secure state does not violate the security permissions 1 = A DMA channel thread in the non-secure state attempts to perform a secure Read or secure Write.</p> <p>This fault is a precise abort.</p>	0x0
ch_periph_err	[6]	R	<p>Indicates if a DMA channel thread, in the nonsecure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions:</p> <p>0 = A DMA channel thread in the Non-secure state does not violate the security permissions 1 = A DMA channel thread in the non-secure state attempts to execute: * DMAWFP to wait for a secure peripheral * DMALDP or DMASTP to notify a secure peripheral * DMAFLUSHP to flush a secure peripheral</p> <p>This fault is a precise abort.</p>	0x0
ch_evnt_err	[5]	R	<p>Indicates if the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions:</p> <p>0 = A DMA channel thread in the non-secure state does not violate the security permissions 1 = A DMA channel thread in the non-secure state attempts to execute either: * DMAWFE to wait for a secure event or * DMASEV to create a secure event or secure interrupt</p> <p>This fault is a precise abort.</p>	0x0

RSVD	[4:2]	-	Reserved	-
operand_invalid	[1]	R	Indicates if the DMA channel thread attempts to execute an instruction operand that is invalid for the DMAC configuration 0 = Valid operand 1 = Invalid operand This fault is a precise abort.	0x0
undef_instr	[0]	R	Indicates if the DMA channel thread was attempts to execute an undefined instruction 0 = Defined instruction 1 = Undefined instruction This fault is a precise abort.	0x0

### 17.5.12 FTR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x44, 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above FTR	[31:0]	R	same as above FTR	0x0

### 17.5.13 FTR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x48, 0x48 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above FTR	[31:0]	R	same as above FTR	0x0

### 17.5.14 FTR3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4C, 0x4C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above FTR	[31:0]	R	same as above FTR	0x0

### 17.5.15 FTR4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x50, 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.16 FTR5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x54, 0x54 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.17 FTR6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x58, 0x58 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.18 FTR7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x5C, 0x5C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.19 CSR0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x100, 0x100 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:22]	-	Reserved	-
CNS	[21]	R	Provides the security of the DMA channel 0 = DMA channel operates in the secure state 1 = DMA channel operates in the non-secure state	0x0
RSVD	[20:16]	-	Reserved	-
dmawfp_periph	[15]	R	When the DMA channel thread executes DMAWFP, this bit indicates if the periph operand is set 0 = Executes DMAWFP with the periph operand not set. 1 = Executes DMAWFP with the periph operand set.	0x0
dmawfp_b_ns	[14]	R	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand is set 0 = Executes DMAWFP with the single operand set 1 = Executes DMAWFP with the burst operand set	0x0
RSVD	[13:9]	-	Reserved	-
Wakeup number	[8:4]	R	When the DMA channel is in the 'Waiting for event' state, or the Waiting for peripheral state, these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is 'Waiting for event', or peripheral, 0 b00001 = DMA channel is 'Waiting for event', or peripheral, 1 b00010 = DMA channel is 'Waiting for event', or peripheral, 2 : b11111 = DMA channel is 'Waiting for event', or peripheral, 31.	0x0
channel status	[3:0]	R	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = "Waiting for event" b0101 = At barrier b0110 = Reserved b0111 = Waiting for peripheral b1000 = Killing b1001 = Completing b1010 to b1101 = Reserved b1110 = Faulting completing b1111 = Faulting	0x0

### 17.5.20 CPC0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x104, 0x104 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
pc_chul	[31:0]	R	Programs the counter for the DMA channel n thread, where n depends on the address of the Register	0x0

### 17.5.21 CSR1

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x108, 0x108 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CSR	[31:0]	R	same as above CSR	0x0

### 17.5.22 CPC1

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x10C, 0x10C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CPC	[31:0]	R	same as above CPC	0x0

### 17.5.23 CSR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x110, 0x110 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CSR	[31:0]	R	same as above CSR	0x0

### 17.5.24 CPC2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x114, 0x114 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CPC	[31:0]	R	same as above CPC	0x0

### 17.5.25 CSR3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x118, 0x118 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CSR	[31:0]	R	same as above CSR	0x0

### 17.5.26 CPC3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x11C, 0x11C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CPC	[31:0]	R	same as above CPC	0x0

### 17.5.27 CSR4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x120, 0x120 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.28 CPC4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x124, 0x124 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.29 CSR5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x128, 0x128 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.30 CPC5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x12C, 0x12C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.31 CSR6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x130, 0x130 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.32 CPC6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x134, 0x134 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.33 CSR7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x138, 0x138 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.34 CPC7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x13C, 0x13C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.35 SAR0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x400, 0x400 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
src_addr	[31:0]	R	Address of the source data for DMA channel n where n depends on the address of the Register	0x0

### 17.5.36 DAR0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x404, 0x404 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
dst_addr	[31:0]	R	Address of the destination data for DMA channel n where n depends on the address of the Register	0x0

### 17.5.37 CCR0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x408, 0x408 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31]	-	Reserved	-
Endian_swap_size	[30:28]	R	b000 = No swap, 8-bit data b001 = Swap bytes within 16-bit data b010 = Swap bytes within 32-bit data b011 = Swap bytes within 64-bit data b100 = Swap bytes within 128-bit data b101 = Reserved b110 = Reserved b111 = Reserved	0x0
dst_cache_ctrl	[27:25]	R	Programs the state of AWCACHE[3, 1:0]a when the DMAC Writes the destination data * Bit[27] 0 = AWCACHE[3] is Low 1 = AWCACHE[3] is High * Bit[26] 0 = AWCACHE[1] is Low 1 = AWCACHE[1] is High * Bit[25] 0 = AWCACHE[0] is Low 1 = AWCACHE[0] is High	0x0
dst_prot_ctrl	[24:22]	R	Programs the state of AWPROT[2:0] when the DMAC Writes the destination data * Bit[24] 0 = AWPROT[2] is Low 1 = AWPROT[2] is High * Bit[23] 0 = AWPROT[1] is Low 1 = AWPROT[1] is High * Bit[22] 0 = AWPROT[0] is Low 1 = AWPROT[0] is High	0x0
dst_burst_len	[21:18]	R	For each burst, these bits program the number of data transfers that the DMAC performs when it Writes the destination data b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers b1111 = 16 data transfers. The total number of bytes that DMAC Writes out of the MFIFO when it executes a DMAST instruction is equal to the product of dst_burst_len and dst_burst_size.	0x0
dst_burst_size	[17:15]	R	For each beat within a burst, it programs the number of bytes that the DMAC Writes to the destination b000 = Writes 1 byte per beat b001 = Writes 2 bytes per beat b010 = Writes 4 bytes per beat b011 = Writes 8 bytes per beat b100 = Writes 16 bytes per beat b101 to b111 = Reserved The total number of bytes that DMAC Writes out of the MFIFO when it executes a DMAST instruction is equal to the product of dst_burst_len and dst_burst_size.	0x0
dst_inc	[14]	R	Programs the burst type that DMAC performs when it Writes the destination data 0 = Fixed-address burst The DMAC signals AWBURST[0] Low 1 = Incrementing-address burst The DMAC signals AWBURST[0] High	0x0

src_cache_ctrl	[13:11]	R	Sets the bits to control the state of ARCACHE[2:0] a when the DMAC reads the source data * Bit[13] 0 = ARCACHE[2] is Low 1 = ARCACHE[2] is High * Bit[12] 0 = ARCACHE[1] is Low 1 = ARCACHE[1] is High * Bit[11] 0 = ARCACHE[0] is Low 1 = ARCACHE[0] is High	0x0
src_prot_ctrl	[10:8]	R	Programs the state of ARPOT[2:0]a when the DMAC Reads the source data * Bit[10] 0 = ARPOT[2] is Low 1 = ARPOT[2] is High * Bit[9] 0 = ARPOT[1] is Low 1 = ARPOT[1] is High * Bit[8] 0 = ARPOT[0] is Low 1 = ARPOT[0] is High	0x0
src_burst_len	[7:4]	R	For each burst, these bits program the number of data transfers that DMAC performs when it Reads the source data b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers : b1111 = 16 data transfers. The total number of bytes that DMAC Reads into the MFIFO when it executes a DMA LD instruction is equal to the product of src_burst_len and src_burst_size.	0x0
src_burst_size	[3:1]	R	For each beat within a burst, src_burst_size programs the number of bytes that the DMAC Reads from the source b000 = Reads 1 byte per beat b001 = Reads 2 bytes per beat b010 = Reads 4 bytes per beat b011 = Reads 8 bytes per beat b100 = Reads 16 bytes per beat b101 to b111 = Reserved. The total number of bytes that the DMAC Reads into the MFIFO when it executes a DMA LD instruction is equal to the product of src_burst_len and src_burst_size.	0x0
src_inc	[0]	R	Programs the burst type that DMAC performs when it reads the source data 0 = Fixed-address burst. The DMAC signals ARBURST[0] Low 1 = Incrementing-address burst. The DMAC signals ARBURST[0] High	0x0

### 17.5.38 LC0\_0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x40C, 0x40C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
Loop counter iterations	[7:0]	R	Number of loop counter iterations	0x0

### 17.5.39 LC1\_0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x410, 0x410 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
Loop counter iterations	[7:0]	R	Number of loop counter iterations	0x0

### 17.5.40 SAR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x420, 0x420 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above SAR	[31:0]	R	same as above SAR	0x0

### 17.5.41 DAR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x424, 0x424 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above DAR	[31:0]	R	same as above DAR	0x0

### 17.5.42 CCR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x428, 0x428 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CCR	[31:0]	R	same as above CCR	0x0

### 17.5.43 LC0\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x42C, 0x42C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above LC0	[31:0]	R	same as above LC0	0x0

### 17.5.44 LC1\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x430, 0x430 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above LC1	[31:0]	R	same as above LC1	0x0

### 17.5.45 SAR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x440, 0x440 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above SAR	[31:0]	R	same as above SAR	0x0

### 17.5.46 DAR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x444, 0x444 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above DAR	[31:0]	R	same as above DAR	0x0

### 17.5.47 CCR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x448, 0x448 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CCR	[31:0]	R	same as above CCR	0x0

### 17.5.48 LC0\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x44C, 0x44C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above LC0	[31:0]	R	same as above LC0	0x0

### 17.5.49 LC1\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x450, 0x450 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above LC1	[31:0]	R	same as above LC1	0x0

### 17.5.50 SAR3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x460, 0x460 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above SAR	[31:0]	R	same as above SAR	0x0

### 17.5.51 DAR3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x464, 0x464 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above DAR	[31:0]	R	same as above DAR	0x0

### 17.5.52 CCR3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x468, 0x468 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above CCR	[31:0]	R	same as above CCR	0x0

### 17.5.53 LC0\_3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x46C, 0x46C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above LC0	[31:0]	R	same as above LC0	0x0

### 17.5.54 LC1\_3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x470, 0x470 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
same as above LC1	[31:0]	R	same as above LC1	0x0

### 17.5.55 SAR4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x480, 0x480 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.56 DAR4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x484, 0x484 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.57 CCR4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x488, 0x488 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.58 LC0\_4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x48C, 0x48C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.59 LC1\_4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x490, 0x490 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.60 SAR5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4A0, 0x4A0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.61 DAR5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4A4, 0x4A4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.62 CCR5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4A8, 0x4A8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.63 LC0\_5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4AC, 0x4AC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.64 LC1\_5

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4B0, 0x4B0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.65 SAR6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4C0, 0x4C0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.66 DAR6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4C4, 0x4C4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.67 CCR6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4C8, 0x4C8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.68 LC0\_6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4CC, 0x4CC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.69 LC1\_6

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4D0, 0x4D0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.70 SAR7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4E0, 0x4E0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.71 DAR7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4E4, 0x4E4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.72 CCR7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4E8, 0x4E8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.73 LC0\_7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4EC, 0x4EC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.74 LC1\_7

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0x4F0, 0x4F0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Not available	[31:0]	R	Not available	0x0

### 17.5.75 DBGSTATUS

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xD00, 0xD00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
dbgstatus	[7:0]	R	The debug status encoding is: 0 = Idle 1 = Busy	0x0

### 17.5.76 DBGCMD

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xD04, 0xD04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
dbgcmd	[1:0]	W	The debug encoding is: b00 = Executes the instruction that contains in the DBGINST[1:0] Registers b01 = Reserved b10 = Reserved b11 = Reserved	0x0

### 17.5.77 DBGINST0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xD08, 0xD08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Instruction byte 1	[31:24]	W	Instruction byte 1 of debug instruction for the DMAC	0x0
Instruction byte 0	[23:16]	W	Instruction byte 0 of debug instruction for the DMAC	0x0
RSVD	[15:11]	-	Reserved	-
Channel number	[10:8]	W	DMA channel number	0x0
RSVD	[7:1]	-	Reserved	-
Debug thread	[0]	W	The debug thread encoding is: 0 = DMA manager thread 1 = DMA channel	0x0

### 17.5.78 DBGINST1

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xD0C, 0xD0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
Instruction byte 5	[31:24]	W	Instruction byte 5 of debug instruction for the DMAC	0x0
Instruction byte 4	[23:16]	W	Instruction byte 4 of debug instruction for the DMAC	0x0
Instruction byte 3	[15:8]	W	Instruction byte 3 of debug instruction for the DMAC	0x0
Instruction byte 2	[7:0]	W	Instruction byte 2 of debug instruction for the DMAC	0x0

### 17.5.79 CR0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE00, 0xE00 Reset Value = 0x003F\_F035

Name	Bit	Type	Description	Reset Value
RSVD	[31:22]	-	Reserved	-
num_events	[21:17]	R	Number of interrupt outputs that DMAC provides b00000 = 1 interrupt output, irq[0] b00001 = 2 interrupt outputs, irq[1:0] b00010 = 3 interrupt outputs, irq[2:0] b11111 = 32 interrupt outputs, irq[31:0].	0x1F
num_periph_req	[16:12]	R	Number of peripheral request interfaces that DMAC provides b00000 = 1 peripheral request interface b00001 = 2 peripheral request interfaces b00010 = 3 peripheral request interfaces b11111 = 32 peripheral request interfaces	0x1F
RSVD	[11:7]	-	Reserved	-
num_chnls	[6:4]	R	Number of DMA channels that DMAC supports b000 = 1 DMA channel b001 = 2 DMA channels b010 = 3 DMA channels b111 = 8 DMA channels	0x3
RSVD	[3]	-	Reserved	-
mgr_ns_at_RST	[2]	R	Indicates the status of the boot_manager_ns signal when DMAC exits from reset 0 = boot_manager_ns is Low 1 = boot_manager_ns is High *NOTE: Reset value of SEC_DMA is 0x0	0x1
Boot_en	[1]	R	Indicates the status of the boot_from_pc signal when the DMAC exits from reset 0 = boot_from_pc is Low 1 = boot_from_pc is High	0x0
periph_req	[0]	R	Supports peripheral requests 0 = DMAC does not provide a peripheral request interface 1 = DMAC provides the number of peripheral request interfaces that the num_periph_req field Specifies	0x1

### 17.5.80 CR1

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE04, 0xE04 Reset Value = 0x0000\_0074

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
num_i-cache_lines	[7:4]	R	Number of i-cache lines b0000 = 1 i-cache line b0001 = 2 i-cache lines b0010 = 3 i-cache lines b1111 = 16 i-cache lines	0x7
RSVD	[3]	-	Reserved	-
i-cache_len	[2:0]	R	The length of an i-cache line b000-b001 = Reserved b010 = 4 bytes b011 = 8 bytes b100 = 16 bytes b101 = 32 bytes b110 to b111 = Reserved	0x4

### 17.5.81 CR2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE08, 0xE08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
boot_addr	[31:0]	R	Provides the value of boot_addr[31:0] when the DMAC exits from reset	0x0

### 17.5.82 CR3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE0C, 0xE0C Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
INS	[31:0]	R	Provides the security state of an event-interrupt resource * Bit[N] 0 = Event<N> or irq[N] is in the secure state * Bit[N] 1 = Event<N> or irq[N] is in the non-secure state *NOTE: Reset value of SEC_DMA is 0x00000000	0xFFFFFFFF

### 17.5.83 CR4

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE10, 0xE10 Reset Value = 0xFFFF\_FFFF

Name	Bit	Type	Description	Reset Value
PNS	[31:0]	R	<p>Provides the security state of the peripheral request interfaces            * Bit[N]            0 = Peripheral request interface, N is in the secure state            * Bit[N]            1 = Peripheral request interface, N is in the nonsecure state            *NOTE: Reset value of SEC_DMA is 0x00000000</p>	0xFFFFFFFF

### 17.5.84 CRD

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE14, 0xE14 Reset Value = 0x03F7\_3732

Name	Bit	Type	Description	Reset Value
RSVD	[31:30]	-	Reserved	-
data_buffer_dep	[29:20]	R	<p>The number of lines that the data buffer contains:            b000000000 = 1 line            b000000001 = 2 lines            b111111111 = 1024 lines</p>	0x3F
rd_q_dep	[19:16]	R	<p>The depth of the Read queue            b0000 = 1 line            b0001 = 2 lines            b1111 = 16 lines</p>	0x7
RSVD	[15]	-	Reserved	-
rd_cap	[14:12]	R	<p>Reads issuing capability that programs the number of outstanding Read transactions            b000 = 1            b001 = 2            b111 = 8</p>	0x3
wr_q_dep	[11:8]	R	<p>The depth of the Write queue            b0000 = 1 line            b0001 = 2 lines            b1111 = 16 lines</p>	0x7
RSVD	[7]	-	Reserved	-
wr_cap	[6:4]	R	<p>Writes issuing capability that programs the number of outstanding Write transactions            b000 = 1            b001 = 2            b111 = 8</p>	0x3
RSVD	[3]	-	Reserved	-
data_width	[2:0]	R	<p>The data bus width of the AXI master interface            b000 = Reserved            b001 = Reserved            b010 = 32-bit            b011 = 64-bit            b100 = 128-bit            b101 to b111 = Reserved</p>	0x2

### 17.5.85 WD

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xE80, 0xE80 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
wd_irq_only	[0]	R/W	Controls the responses of DMAC when it detects a lock-up condition 0 = DMAC aborts all the contributing DMA channels and sets irq_abort High 1 = DMAC sets irq_abort High	0x0

### 17.5.86 periph\_id\_0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFE0, 0xFE0 Reset Value = 0x0000\_0030

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
part_number_0	[7:0]	R	Returns 0x30	0x30

### 17.5.87 periph\_id\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFE4, 0xFE4 Reset Value = 0x0000\_0013

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
designer_0	[7:4]	R	Returns 0x1	0x1
part_number_1	[3:0]	R	Returns 0x3	0x3

### 17.5.88 periph\_id\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFE8, 0xFE8 Reset Value = 0x0000\_0034

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
revision	[7:4]	R	Returns 0x3 (for r1p2)	0x3
designer_1	[3:0]	R	Returns 0x4	0x4

### 17.5.89 periph\_id\_3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFEC, 0xFEC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
RSVD	[7:1]	-	Reserved	-
part_number_1	[0]	R	Returns 0x0 to indicate that the DMAC does not contain integration test logic	0x0

### 17.5.90 pcell\_id\_0

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFF0, 0xFF0 Reset Value = 0x0000\_000D

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
pcell_id_0	[7:0]	R	Returns 0x0D	0x0D

### 17.5.91 pcell\_id\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFF4, 0xFF4 Reset Value = 0x0000\_00F0

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
pcell_id_1	[7:0]	R	Returns 0xF0	0xF0

### 17.5.92 pcell\_id\_2

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFF8, 0xFF8 Reset Value = 0x0000\_0005

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
pcell_id_2	[7:0]	R	Returns 0x05	0x05

### 17.5.93 pcell\_id\_3

- Base Address: 2045\_0000(PDMA0)
- Base Address: 2046\_0000(PDMA1)
- Address = Base Address + 0xFFC, 0xFFC Reset Value = 0x0000\_00B1

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
pcell_id_3	[7:0]	R	Returns 0xB1	0xB1

# 18 GPIO

## 18.1 Overview

### 18.1.1 Block Diagram

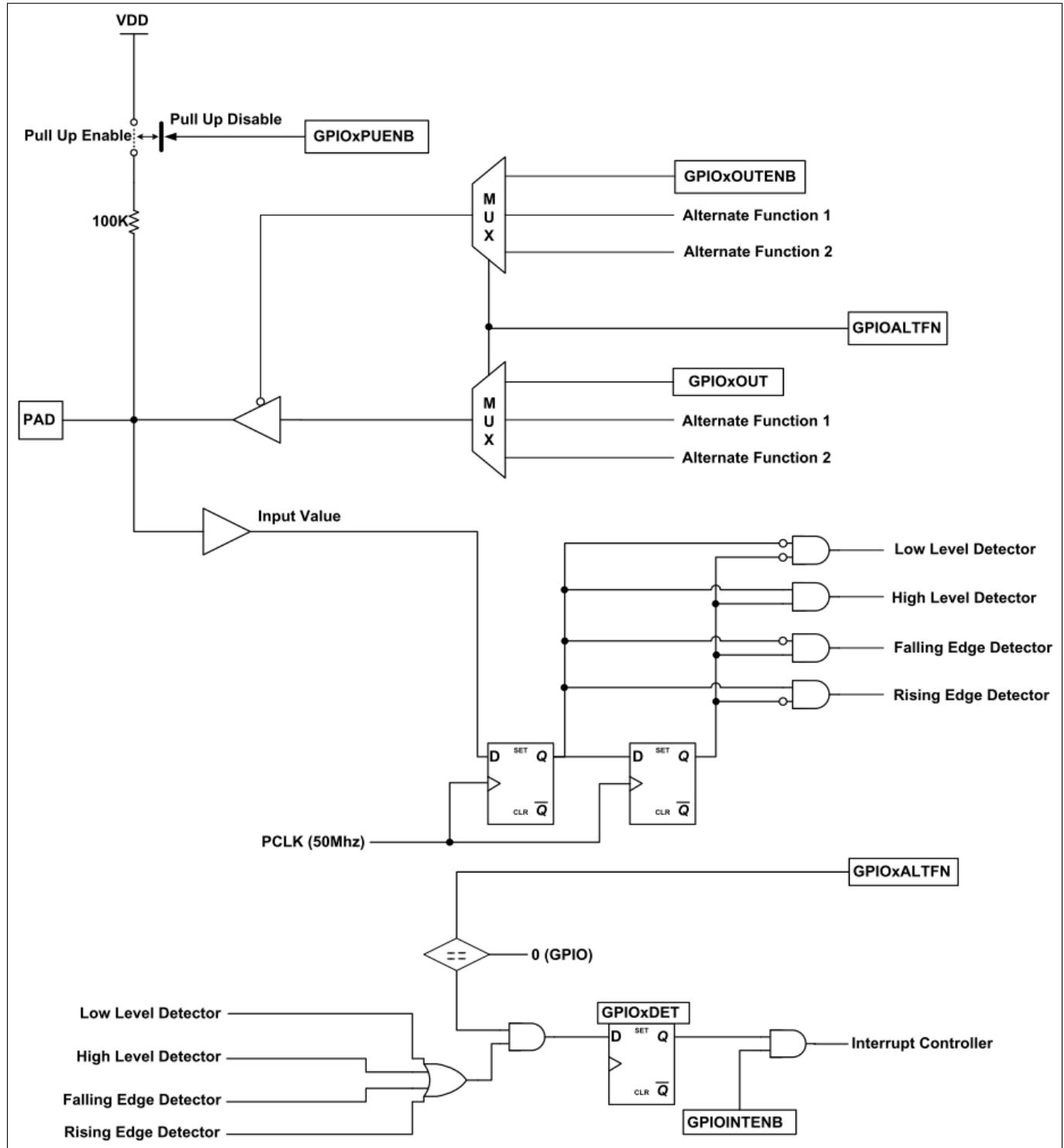


Figure 18.1: GPIO Block Diagram

## 18.2 Functional Description

DRONE\_SOC GPIO Pins have an internal pull-up resistance of 100K ohm. The current of the pull-up resistance (for VDD = 3.3 V and V (PAD) = 0V) is listed in the table below:

Pull up	MIN	TYP	MAX	UNIT
ENABLE	10	33	72	uA
DISABLE	-	-	0.1	uA

**Table 18.1:** Pull-Up Resister Current

Most DRONE\_SOC GPIO ports contain the Alternate function (some ports supports up to Alternate Function2). All GPIO ports should be set as GPIO function or Alternate Function suitable for the user's purposes and this setting can easily be performed with GPIO registers. In addition, all GPIO pull-up resistances are enabled/ disabled. This setting operates when the system is fully booted and does not affect the system in initial booting. If there is a value which needs to be determined in system booting, the value is given by inserting a pull up/down resistance from outside.

### 18.2.1 Input Operation

To use GPIO for input, the GPIO function should be selected by setting the relevant bit of the GPIO Alternate Function Select register as b'00 to select the GPIO function. Inaddition, the GPIO Input mode should be, also, selected by the GPIOx Output Enableregister (GPIOxOUTENB) as '0'.

An input signal is detected by selecting a desired detection type with the GPIOx EventDetect Mode register. Four types of input signal can be detected: Low Level, High Level,Falling edge and Rising edge. The GPIOx Event Detect Mode registers consist of GPIOxEvent Detect Mode register0 (GPIOxDETMODE0) and GPIOx Event Detect Moderegister1 (GPIOxDETMODE1).

To use interrupt, set the GPIOx Interrupt Enable Register (GPIOxINTENB) as '1'.

The GPIOx Event Detect Register (GPIOxDET) enables checking the generation of anevent via GPIO and may be used as the Pending Clear function when an interrupt occurs. When the GPIOx PAD Status Register (GPIOxPAD) is set as GPIO Input mode, the levelof the relevant GPIOx PAD can be checked.

The Open drain pins operate in Input mode only when theGPIOx Output register (GPIOxOUT) is set as '1'. The Open drain pins are operated by the GPIOx Output Register (GPIOxOUT) even if the GPIOx Output Enable register(GPIOxOUTENB) is set as Input mode.

### 18.2.2 Output Operation

To use GPIO for output, the GPIO function should be selected by setting the relevant bit of the GPIOx. Alternate Function Select register should be set as b'00 to select the GPIOfunction. In addition, the GPIOx Output mode should also be selected by setting theGPIOx Output Enable register as '1'.

If you set a desired output value (low level: '0', high level: '1') with the GPIOx OutputRegister (GPIOxOUT), the value is reflected to the corresponding bit.

The Open drain pins operate in output mode only when theGPIOx Output register (GPIOxOUT) is set as '0'. The Open drain pins are operated by theGPIOx Output Register (GPIOxOUT) even if the GPIOx Output Enable register(GPIOxOUTENB) is set as Input mode.

### 18.2.3 Alternate Function Operation

Among the 226 GPIO pins of the DRONE\_SOC, most GPIO pins have an Alternate function. However, the Alternate function and the GPIO function should not be used simultaneously. Therefore, Alternate Function1 and Alternate Function2 are operated by setting the corresponding bits of the GPIOx Alternate Function Select register as b'01 and b'10, respectively.

### 18.2.4 Secure / Non-secure Operation

DRONE\_SOC GPIO support secure access by dividing APB interface to secure APB and non-secure APB. Secure CPU controls secure APB and non-secure CPU controls non-secure APB.

These are the GPIO access policy.

1. Secure APB can access all GPIO register.
2. Non-secure APB can access only non-secure register.
3. If non-secure APB access to secure register,
  - read : non-secure APB just read 0, instead of register value.
  - write : non-secure APB cannot write to secure register
4. GPIO uses the Secure Marking Register (32bit) to mark the relevant register is secure or non-secure.
  - 1 : secure
  - 0 : non-secure
  - default : 0 (non-secure)

Only secure CPU can set or clear Secure Marking Register.

## 18.3 GPIO Register Description

### 18.3.1 Register Map Summary

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])

Register	Offset	Description	Reset Value
GPIOxOUT	0x00	GPIOx OUTPUT REGISTER	0x0000_0000
GPIOxOUTENB	0x04	GPIOx OUTPUT ENABLE REGISTER	0x0000_0000
GPIOxDETMODE0	0x08	GPIOx EVENT DETECT MODE REGISTER 0	0x0000_0000
GPIOxDETMODE1	0x0C	GPIOx EVENT DETECT MODE REGISTER 1	0x0000_0000
GPIOxINTENB	0x10	GPIOx INTERRUPT ENABLE REGISTER	0x0000_0000
GPIOxDET	0x14	GPIOx EVENT DETECT REGISTER	0x0000_0000
GPIOxPAD	0x18	GPIOx PAD STATUS REGISTER	0x0000_0000
GPIOxALTFN0	0x20	GPIOx ALTERNATE FUNCTION SELECT REGISTER 0	0x0000_0000
GPIOxALTFN1	0x2C	GPIOx ALTERNATE FUNCTION SELECT REGISTER 1	0x0000_0000
GPIOxDETMODEEX	0x30	GPIOx EVENT DETECT MODE EXTENDED REGISTER	0x0000_0000
GPIOxDETENB	0x3C	GPIOx DETECT ENABLE REGISTER	0x0000_0000
GPIOx_SLEW	0x40	GPIOx SLEW REGISTER	0x0000_0000
GPIOx_SLEW_DISABLE_DEFAULT	0x44	GPIOx SLEW DISABLEDEFAULTREGISTER	0x0000_0000
GPIOx_DRV1	0x48	GPIOx DRV1 REGISTER	0x0000_0000
GPIOx_DRV1_DISABLE_DEFAULT	0x4C	GPIOx DRV1 DISABLEDEFAULT REGISTER	0x0000_0000
GPIOx_DRV0	0x50	GPIOx DRV0 REGISTER	0x0000_0000
GPIOx_DRV0_DISABLE_DEFAULT	0x54	GPIOx DRV0 DISABLEDEFAULT REGISTER	0x0000_0000
GPIOx_PULLSEL	0x58	GPIOx PULLSEL REGISTER	0x0000_0000
GPIOx_PULLSEL_DISABLE_DEFAULT	0x5C	GPIOx PULLSEL DISABLEDEFAULT REGISTER	0x0000_0000
GPIOx_PULLENB	0x60	GPIOx PULLENB REGISTER	0x0000_0000
GPIOx_PULLENB_DISABLE_DEFAULT	0x64	GPIOx PULLENB DISABLEDEFAULT REGISTER	0x0000_0000
GPIOx_SECURE_MARKING_REGISTER	0x70		0x0000_0000
GPIOx_INPUTENB	0x74	GPIOx INPUTENB REGISTER	0x0000_0000
GPIOx_INPUTENB_DISABLE_DEFAULT	0x78	GPIOx INPUTENB DISABLEDEFAULT REGISTER	0x0000_0000

### 18.3.2 GPIOxOUT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXOUT	[31:0]	R/W	GPIOx[31:0]: Specifies the output value in GPIOx output mode. Note: This bit should be set as 0 (Input mode) or 1 (Output mode) to use the Open drain pins in Input/Output mode. 0:Low Level 1:High Level	0

### 18.3.3 GPIOxOUTENB

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXOUTENB	[31:0]	R/W	GPIOx[31:0]: Specifies GPIOx In/Out mode. Note: The Open drain pins are operated in Input/Output mode by the GPIOxOUTPUT register (GPIOxOUT) and not by this bit. 0:Input Mode 1:Output Mode	0

### 18.3.4 GPIOxDETMODE0

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXDETMODE0_15	[31:30]	R/W	<p>Specifies Detect mode when GPIOx15 is in Input mode.            It is configured the combination of GPIOXDET_EX15(1 bit) + GPIOXDETMODE0_15(2 bit.).            Considers GPIOXDET_EX15 as well as GPIOXDETMODE0_15.            Note)            - First bit is GPIOXDET_EX15            - Second and third bit is GPIOXDETMODE0_15            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</p>	0
GPIOXDETMODE0_14	[29:28]	R/W	<p>Specifies Detect mode when GPIOx14 is in Input mode.            It is configured the combination of GPIOXDET_EX14(1 bit) + GPIOXDETMODE0_14(2 bit.).            Considers GPIOXDET_EX14 as well as GPIOXDETMODE0_14.            Note)            - First bit is GPIOXDET_EX14            - Second and third bit is GPIOXDETMODE0_14            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</p>	0
GPIOXDETMODE0_13	[27:26]	R/W	<p>Specifies Detect mode when GPIOx13 is in Input mode.            It is configured the combination of GPIOXDET_EX13(1 bit) + GPIOXDETMODE0_13(2 bit.).            Considers GPIOXDET_EX13 as well as GPIOXDETMODE0_13.            Note)            - First bit is GPIOXDET_EX13            - Second and third bit is GPIOXDETMODE0_13            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</p>	0
GPIOXDETMODE0_12	[25:24]	R/W	<p>Specifies Detect mode when GPIOx12 is in Input mode.            It is configured the combination of GPIOXDET_EX12(1 bit) + GPIOXDETMODE0_12(2 bit.).            Considers GPIOXDET_EX12 as well as GPIOXDETMODE0_12.            Note)            - First bit is GPIOXDET_EX12            - Second and third bit is GPIOXDETMODE0_12            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</p>	0
GPIOXDETMODE0_11	[23:22]	R/W	<p>Specifies Detect mode when GPIOx11 is in Input mode.            It is configured the combination of GPIOXDET_EX11(1 bit) + GPIOXDETMODE0_11(2 bit.).            Considers GPIOXDET_EX11 as well as GPIOXDETMODE0_11.            Note)            - First bit is GPIOXDET_EX11            - Second and third bit is GPIOXDETMODE0_11            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</p>	0

GPIOXDETMODE0_10	[21:20]	R/W	<p>Specifies Detect mode when GPIOx10 is in Input mode. It is configured the combination of GPIOXDET_EX10(1 bit) + GPIOXDETMODE0_10(2 bit.). Considers GPIOXDET_EX10 as well as GPIOXDETMODE0_10. Note)            - First bit is GPIOXDET_EX10            - Second and third bit is GPIOXDETMODE0_10            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_9	[19:18]	R/W	<p>Specifies Detect mode when GPIOx9 is in Input mode. It is configured the combination of GPIOXDET_EX9(1 bit) + GPIOXDETMODE0_9(2 bit.). Considers GPIOXDET_EX9 as well as GPIOXDETMODE0_9. Note)            - First bit is GPIOXDET_EX9            - Second and third bit is GPIOXDETMODE0_9            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_8	[17:16]	R/W	<p>Specifies Detect mode when GPIOx8 is in Input mode. It is configured the combination of GPIOXDET_EX8(1 bit) + GPIOXDETMODE0_8(2 bit.). Considers GPIOXDET_EX8 as well as GPIOXDETMODE0_8. Note)            - First bit is GPIOXDET_EX8            - Second and third bit is GPIOXDETMODE0_8            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_7	[15:14]	R/W	<p>Specifies Detect mode when GPIOx7 is in Input mode. It is configured the combination of GPIOXDET_EX7(1 bit) + GPIOXDETMODE0_7(2 bit.). Considers GPIOXDET_EX7 as well as GPIOXDETMODE0_7. Note)            - First bit is GPIOXDET_EX7            - Second and third bit is GPIOXDETMODE0_7            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_6	[13:12]	R/W	<p>Specifies Detect mode when GPIOx6 is in Input mode. It is configured the combination of GPIOXDET_EX6(1 bit) + GPIOXDETMODE0_6(2 bit.). Considers GPIOXDET_EX6 as well as GPIOXDETMODE0_6. Note)            - First bit is GPIOXDET_EX6            - Second and third bit is GPIOXDETMODE0_6            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_5	[11:10]	R/W	<p>Specifies Detect mode when GPIOx5 is in Input mode. It is configured the combination of GPIOXDET_EX5(1 bit) + GPIOXDETMODE0_5(2 bit.). Considers GPIOXDET_EX5 as well as GPIOXDETMODE0_5. Note)            - First bit is GPIOXDET_EX5            - Second and third bit is GPIOXDETMODE0_5            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_4	[9:8]	R/W	<p>Specifies Detect mode when GPIOx4 is in Input mode. It is configured the combination of GPIOXDET_EX4(1 bit) + GPIOXDETMODE0_4(2 bit.). Considers GPIOXDET_EX4 as well as GPIOXDETMODE0_4. Note)            - First bit is GPIOXDET_EX4            - Second and third bit is GPIOXDETMODE0_4            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

GPIOXDETMODE0_3	[7:6]	R/W	<p>Specifies Detect mode when GPIOx3 is in Input mode. It is configured the combination of GPIOXDET_EX3(1 bit) + GPIOXDETMODE0_3(2 bit). Considers GPIOXDET_EX3 as well as GPIOXDETMODE0_3. Note)            - First bit is GPIOXDET_EX3            - Second and third bit is GPIOXDETMODE0_3            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_2	[5:4]	R/W	<p>Specifies Detect mode when GPIOx2 is in Input mode. It is configured the combination of GPIOXDET_EX2(1 bit) + GPIOXDETMODE0_2(2 bit). Considers GPIOXDET_EX2 as well as GPIOXDETMODE0_2. Note)            - First bit is GPIOXDET_EX2            - Second and third bit is GPIOXDETMODE0_2            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_1	[3:2]	R/W	<p>Specifies Detect mode when GPIOx1 is in Input mode. It is configured the combination of GPIOXDET_EX1(1 bit) + GPIOXDETMODE0_1(2 bit). Considers GPIOXDET_EX1 as well as GPIOXDETMODE0_1. Note)            - First bit is GPIOXDET_EX1            - Second and third bit is GPIOXDETMODE0_1            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE0_0	[1:0]	R/W	<p>Specifies Detect mode when GPIOx0 is in Input mode. It is configured the combination of GPIOXDET_EX0(1 bit) + GPIOXDETMODE0_0(2 bit). Considers GPIOXDET_EX0 as well as GPIOXDETMODE0_0. Note)            - First bit is GPIOXDET_EX0            - Second and third bit is GPIOXDETMODE0_0            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

### 18.3.5 GPIOxDETMODE1

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
------	-----	------	-------------	-------------

GPIOXDETMODE1_31	[31:30]	R/W	<p>Specifies Detect mode when GPIOx31 is in Input mode. It is configured the combination of GPIOXDET_EX31(1 bit) + GPIOXDETMODE1_31(2 bit.). Considers GPIOXDET_EX31 as well as GPIOXDETMODE1_31. Note)            - First bit is GPIOXDET_EX31            - Second and third bit is GPIOXDETMODE1_31            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_30	[29:28]	R/W	<p>Specifies Detect mode when GPIOx30 is in Input mode. It is configured the combination of GPIOXDET_EX30(1 bit) + GPIOXDETMODE1_30(2 bit.). Considers GPIOXDET_EX30 as well as GPIOXDETMODE1_30. Note)            - First bit is GPIOXDET_EX30            - Second and third bit is GPIOXDETMODE1_30            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_29	[27:26]	R/W	<p>Specifies Detect mode when GPIOx29 is in Input mode. It is configured the combination of GPIOXDET_EX29(1 bit) + GPIOXDETMODE1_29(2 bit.). Considers GPIOXDET_EX29 as well as GPIOXDETMODE1_29. Note)            - First bit is GPIOXDET_EX29            - Second and third bit is GPIOXDETMODE1_29            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_28	[25:24]	R/W	<p>Specifies Detect mode when GPIOx28 is in Input mode. It is configured the combination of GPIOXDET_EX28(1 bit) + GPIOXDETMODE1_28(2 bit.). Considers GPIOXDET_EX28 as well as GPIOXDETMODE1_28. Note)            - First bit is GPIOXDET_EX28            - Second and third bit is GPIOXDETMODE1_28            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_27	[23:22]	R/W	<p>Specifies Detect mode when GPIOx27 is in Input mode. It is configured the combination of GPIOXDET_EX27(1 bit) + GPIOXDETMODE1_27(2 bit.). Considers GPIOXDET_EX27 as well as GPIOXDETMODE1_27. Note)            - First bit is GPIOXDET_EX27            - Second and third bit is GPIOXDETMODE1_27            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_26	[21:20]	R/W	<p>Specifies Detect mode when GPIOx26 is in Input mode. It is configured the combination of GPIOXDET_EX26(1 bit) + GPIOXDETMODE1_26(2 bit.). Considers GPIOXDET_EX26 as well as GPIOXDETMODE1_26. Note)            - First bit is GPIOXDET_EX26            - Second and third bit is GPIOXDETMODE1_26            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_25	[19:18]	R/W	<p>Specifies Detect mode when GPIOx25 is in Input mode. It is configured the combination of GPIOXDET_EX25(1 bit) + GPIOXDETMODE1_25(2 bit.). Considers GPIOXDET_EX25 as well as GPIOXDETMODE1_25. Note)            - First bit is GPIOXDET_EX25            - Second and third bit is GPIOXDETMODE1_25            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

GPIOXDETMODE1_24	[17:16]	R/W	<p>Specifies Detect mode when GPIOx24 is in Input mode. It is configured the combination of GPIOXDET_EX24(1 bit) + GPIOXDETMODE1_24(2 bit.). Considers GPIOXDET_EX24 as well as GPIOXDETMODE1_24. Note)            - First bit is GPIOXDET_EX24            - Second and third bit is GPIOXDETMODE1_24            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_23	[15:14]	R/W	<p>Specifies Detect mode when GPIOx23 is in Input mode. It is configured the combination of GPIOXDET_EX23(1 bit) + GPIOXDETMODE1_23(2 bit.). Considers GPIOXDET_EX23 as well as GPIOXDETMODE1_23. Note)            - First bit is GPIOXDET_EX23            - Second and third bit is GPIOXDETMODE1_23            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_22	[13:12]	R/W	<p>Specifies Detect mode when GPIOx22 is in Input mode. It is configured the combination of GPIOXDET_EX22(1 bit) + GPIOXDETMODE1_22(2 bit.). Considers GPIOXDET_EX22 as well as GPIOXDETMODE1_22. Note)            - First bit is GPIOXDET_EX22            - Second and third bit is GPIOXDETMODE1_22            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_21	[11:10]	R/W	<p>Specifies Detect mode when GPIOx21 is in Input mode. It is configured the combination of GPIOXDET_EX21(1 bit) + GPIOXDETMODE1_21(2 bit.). Considers GPIOXDET_EX21 as well as GPIOXDETMODE1_21. Note)            - First bit is GPIOXDET_EX21            - Second and third bit is GPIOXDETMODE1_21            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_20	[9:8]	R/W	<p>Specifies Detect mode when GPIOx20 is in Input mode. It is configured the combination of GPIOXDET_EX20(1 bit) + GPIOXDETMODE1_20(2 bit.). Considers GPIOXDET_EX20 as well as GPIOXDETMODE1_20. Note)            - First bit is GPIOXDET_EX20            - Second and third bit is GPIOXDETMODE1_20            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_19	[7:6]	R/W	<p>Specifies Detect mode when GPIOx19 is in Input mode. It is configured the combination of GPIOXDET_EX19(1 bit) + GPIOXDETMODE1_19(2 bit.). Considers GPIOXDET_EX19 as well as GPIOXDETMODE1_19. Note)            - First bit is GPIOXDET_EX19            - Second and third bit is GPIOXDETMODE1_19            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_18	[5:4]	R/W	<p>Specifies Detect mode when GPIOx18 is in Input mode. It is configured the combination of GPIOXDET_EX18(1 bit) + GPIOXDETMODE1_18(2 bit.). Considers GPIOXDET_EX18 as well as GPIOXDETMODE1_18. Note)            - First bit is GPIOXDET_EX18            - Second and third bit is GPIOXDETMODE1_18            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

GPIOXDETMODE1_17	[3:2]	R/W	<p>Specifies Detect mode when GPIOx17 is in Input mode. It is configured the combination of GPIOXDET_EX17(1 bit) + GPIOXDETMODE1_17(2 bit.). Considers GPIOXDET_EX17 as well as GPIOXDETMODE1_17. Note)            - First bit is GPIOXDET_EX17            - Second and third bit is GPIOXDETMODE1_17            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDETMODE1_16	[1:0]	R/W	<p>Specifies Detect mode when GPIOx16 is in Input mode. It is configured the combination of GPIOXDET_EX16(1 bit) + GPIOXDETMODE1_16(2 bit.). Considers GPIOXDET_EX16 as well as GPIOXDETMODE1_16. Note)            - First bit is GPIOXDET_EX16            - Second and third bit is GPIOXDETMODE1_16            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

### 18.3.6 GPIOxINTENB

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXINTENB	[31:0]	R/W	GPIOx[31:0]: Specifies the use of an interrupt when a GPIOx Event occurs. The events specified in GPIOxDETMODE0 and GPIOxDETMODE1 are used. 0 : Disable 1 : Enable	0

### 18.3.7 GPIOxDET

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])

- Address = Base Address + 0x14 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXDET	[31:0]	R/W	GPIOx[31:0]: Shows if an event is detected in accordance with Event Detect mode in GPIOx Input Mode. Set 1 to clear the relevant bit. GPIOx[31:0] is used as a Pending register when an interrupt occurs. Read : 0 : Not Detect 1 : Detected Write : 0: Not Clear 1: Clear	0

### 18.3.8 GPIOxPAD

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXPAD	[31:0]	R	GPIOx[31:0]: Can read the level of PAD in GPIOx Input mode. The data read in this register is the data not passing a filter and reflects the PAD status itself. 0 : Low Level 1 : High Level	:no reset

### 18.3.9 GPIOxALTFN0

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXALTFN0_15	[31:30]	R/W	GPIOx[15]: Selects the function of GPIOx 15pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0

GPIOXALTFN0_14	[29:28]	R/W	GPIOx[14]: Selects the function of GPIOx 14pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_13	[27:26]	R/W	GPIOx[13]: Selects the function of GPIOx 13pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_12	[25:24]	R/W	GPIOx[12]: Selects the function of GPIOx 12pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_11	[23:22]	R/W	GPIOx[11]: Selects the function of GPIOx 11pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_10	[21:20]	R/W	GPIOx[10]: Selects the function of GPIOx 10pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_9	[19:18]	R/W	GPIOx[9]: Selects the function of GPIOx 9pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_8	[17:16]	R/W	GPIOx[8]: Selects the function of GPIOx 8pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_7	[15:14]	R/W	GPIOx[7]: Selects the function of GPIOx 7pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_6	[13:12]	R/W	GPIOx[6]: Selects the function of GPIOx 6pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_5	[11:10]	R/W	GPIOx[5]: Selects the function of GPIOx 5pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_4	[9:8]	R/W	GPIOx[4]: Selects the function of GPIOx 4pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_3	[7:6]	R/W	GPIOx[3]: Selects the function of GPIOx 3pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_2	[5:4]	R/W	GPIOx[2]: Selects the function of GPIOx 2pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_1	[3:2]	R/W	GPIOx[1]: Selects the function of GPIOx 1pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN0_0	[1:0]	R/W	GPIOx[0]: Selects the function of GPIOx 0pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0

### 18.3.10 GPIOxALTFN1

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x2C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXALTFN1_31	[31:30]	R/W	GPIOx[31]: Selects the function of GPIOx 31pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_30	[29:28]	R/W	GPIOx[30]: Selects the function of GPIOx 30pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_29	[27:26]	R/W	GPIOx[29]: Selects the function of GPIOx 29pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_28	[25:24]	R/W	GPIOx[28]: Selects the function of GPIOx 28pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_27	[23:22]	R/W	GPIOx[27]: Selects the function of GPIOx 27pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_26	[21:20]	R/W	GPIOx[26]: Selects the function of GPIOx 26pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_25	[19:18]	R/W	GPIOx[25]: Selects the function of GPIOx 25pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_24	[17:16]	R/W	GPIOx[24]: Selects the function of GPIOx 24pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_23	[15:14]	R/W	GPIOx[23]: Selects the function of GPIOx 23pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_22	[13:12]	R/W	GPIOx[22]: Selects the function of GPIOx 22pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_21	[11:10]	R/W	GPIOx[21]: Selects the function of GPIOx 21pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_20	[9:8]	R/W	GPIOx[20]: Selects the function of GPIOx 20pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_19	[7:6]	R/W	GPIOx[19]: Selects the function of GPIOx 19pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_18	[5:4]	R/W	GPIOx[18]: Selects the function of GPIOx 18pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_17	[3:2]	R/W	GPIOx[17]: Selects the function of GPIOx 17pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0
GPIOXALTFN1_16	[1:0]	R/W	GPIOx[16]: Selects the function of GPIOx 16pin. 00 : ALT Function0 01 : ALT Function1 10 : ALT Function2 11 : ALT Function3	0

### 18.3.11 GPIOxDETMODEEX

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])

- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXDET_EX31	[31]	R/W	<p>Specifies Detect mode when GPIOx31 is in Input mode. It is configured the combination of GPIOXDET_EX31(1 bit) + GPIOXDETMODE1_31(2 bit.). Considers GPIOXDET_EX31 as well as GPIOXDETMODE1_31.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX31</li> <li>- Second and third bit is GPIOXDETMODE1_31</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0
GPIOXDET_EX30	[30]	R/W	<p>Specifies Detect mode when GPIOx30 is in Input mode. It is configured the combination of GPIOXDET_EX30(1 bit) + GPIOXDETMODE1_30(2 bit.). Considers GPIOXDET_EX30 as well as GPIOXDETMODE1_30.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX30</li> <li>- Second and third bit is GPIOXDETMODE1_30</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0
GPIOXDET_EX29	[29]	R/W	<p>Specifies Detect mode when GPIOx29 is in Input mode. It is configured the combination of GPIOXDET_EX29(1 bit) + GPIOXDETMODE1_29(2 bit.). Considers GPIOXDET_EX29 as well as GPIOXDETMODE1_29.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX29</li> <li>- Second and third bit is GPIOXDETMODE1_29</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0
GPIOXDET_EX28	[28]	R/W	<p>Specifies Detect mode when GPIOx28 is in Input mode. It is configured the combination of GPIOXDET_EX28(1 bit) + GPIOXDETMODE1_28(2 bit.). Considers GPIOXDET_EX28 as well as GPIOXDETMODE1_28.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX28</li> <li>- Second and third bit is GPIOXDETMODE1_28</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0
GPIOXDET_EX27	[27]	R/W	<p>Specifies Detect mode when GPIOx27 is in Input mode. It is configured the combination of GPIOXDET_EX27(1 bit) + GPIOXDETMODE1_27(2 bit.). Considers GPIOXDET_EX27 as well as GPIOXDETMODE1_27.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX27</li> <li>- Second and third bit is GPIOXDETMODE1_27</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0
GPIOXDET_EX26	[26]	R/W	<p>Specifies Detect mode when GPIOx26 is in Input mode. It is configured the combination of GPIOXDET_EX26(1 bit) + GPIOXDETMODE1_26(2 bit.). Considers GPIOXDET_EX26 as well as GPIOXDETMODE1_26.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX26</li> <li>- Second and third bit is GPIOXDETMODE1_26</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0
GPIOXDET_EX25	[25]	R/W	<p>Specifies Detect mode when GPIOx25 is in Input mode. It is configured the combination of GPIOXDET_EX25(1 bit) + GPIOXDETMODE1_25(2 bit.). Considers GPIOXDET_EX25 as well as GPIOXDETMODE1_25.</p> <p>(Note)</p> <ul style="list-style-type: none"> <li>- First bit is GPIOXDET_EX25</li> <li>- Second and third bit is GPIOXDETMODE1_25</li> <li>000 : Low Level 001 : High Level</li> <li>010 : Falling Edge 011 : Rising Edge</li> <li>100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved</li> </ul>	0

GPIOXDET_EX24	[24]	R/W	<p>Specifies Detect mode when GPIOx24 is in Input mode. It is configured the combination of GPIOXDET_EX24(1 bit) + GPIOXDETMODE1_24(2 bit.). Considers GPIOXDET_EX24 as well as GPIOXDETMODE1_24. Note)            - First bit is GPIOXDET_EX24            - Second and third bit is GPIOXDETMODE1_24            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX23	[23]	R/W	<p>Specifies Detect mode when GPIOx23 is in Input mode. It is configured the combination of GPIOXDET_EX23(1 bit) + GPIOXDETMODE1_23(2 bit.). Considers GPIOXDET_EX23 as well as GPIOXDETMODE1_23. Note)            - First bit is GPIOXDET_EX23            - Second and third bit is GPIOXDETMODE1_23            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX22	[22]	R/W	<p>Specifies Detect mode when GPIOx22 is in Input mode. It is configured the combination of GPIOXDET_EX22(1 bit) + GPIOXDETMODE1_22(2 bit.). Considers GPIOXDET_EX22 as well as GPIOXDETMODE1_22. Note)            - First bit is GPIOXDET_EX22            - Second and third bit is GPIOXDETMODE1_22            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX21	[21]	R/W	<p>Specifies Detect mode when GPIOx21 is in Input mode. It is configured the combination of GPIOXDET_EX21(1 bit) + GPIOXDETMODE1_21(2 bit.). Considers GPIOXDET_EX21 as well as GPIOXDETMODE1_21. Note)            - First bit is GPIOXDET_EX21            - Second and third bit is GPIOXDETMODE1_21            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX20	[20]	R/W	<p>Specifies Detect mode when GPIOx20 is in Input mode. It is configured the combination of GPIOXDET_EX20(1 bit) + GPIOXDETMODE1_20(2 bit.). Considers GPIOXDET_EX20 as well as GPIOXDETMODE1_20. Note)            - First bit is GPIOXDET_EX20            - Second and third bit is GPIOXDETMODE1_20            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX19	[19]	R/W	<p>Specifies Detect mode when GPIOx19 is in Input mode. It is configured the combination of GPIOXDET_EX19(1 bit) + GPIOXDETMODE1_19(2 bit.). Considers GPIOXDET_EX19 as well as GPIOXDETMODE1_19. Note)            - First bit is GPIOXDET_EX19            - Second and third bit is GPIOXDETMODE1_19            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX18	[18]	R/W	<p>Specifies Detect mode when GPIOx18 is in Input mode. It is configured the combination of GPIOXDET_EX18(1 bit) + GPIOXDETMODE1_18(2 bit.). Considers GPIOXDET_EX18 as well as GPIOXDETMODE1_18. Note)            - First bit is GPIOXDET_EX18            - Second and third bit is GPIOXDETMODE1_18            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

GPIOXDET_EX17	[17]	R/W	<p>Specifies Detect mode when GPIOx17 is in Input mode. It is configured the combination of GPIOXDET_EX17(1 bit) + GPIOXDETMODE1_17(2 bit.). Considers GPIOXDET_EX17 as well as GPIOXDETMODE1_17. Note)            - First bit is GPIOXDET_EX17            - Second and third bit is GPIOXDETMODE1_17            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX16	[16]	R/W	<p>Specifies Detect mode when GPIOx16 is in Input mode. It is configured the combination of GPIOXDET_EX16(1 bit) + GPIOXDETMODE1_16(2 bit.). Considers GPIOXDET_EX16 as well as GPIOXDETMODE1_16. Note)            - First bit is GPIOXDET_EX16            - Second and third bit is GPIOXDETMODE1_16            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX15	[15]	R/W	<p>Specifies Detect mode when GPIOx15 is in Input mode. It is configured the combination of GPIOXDET_EX15(1 bit) + GPIOXDETMODE0_15(2 bit.). Considers GPIOXDET_EX15 as well as GPIOXDETMODE0_15. Note)            - First bit is GPIOXDET_EX15            - Second and third bit is GPIOXDETMODE0_15            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX14	[14]	R/W	<p>Specifies Detect mode when GPIOx14 is in Input mode. It is configured the combination of GPIOXDET_EX14(1 bit) + GPIOXDETMODE0_14(2 bit.). Considers GPIOXDET_EX14 as well as GPIOXDETMODE0_14. Note)            - First bit is GPIOXDET_EX14            - Second and third bit is GPIOXDETMODE0_14            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX13	[13]	R/W	<p>Specifies Detect mode when GPIOx13 is in Input mode. It is configured the combination of GPIOXDET_EX13(1 bit) + GPIOXDETMODE0_13(2 bit.). Considers GPIOXDET_EX13 as well as GPIOXDETMODE0_13. Note)            - First bit is GPIOXDET_EX13            - Second and third bit is GPIOXDETMODE0_13            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX12	[12]	R/W	<p>Specifies Detect mode when GPIOx12 is in Input mode. It is configured the combination of GPIOXDET_EX12(1 bit) + GPIOXDETMODE0_12(2 bit.). Considers GPIOXDET_EX12 as well as GPIOXDETMODE0_12. Note)            - First bit is GPIOXDET_EX12            - Second and third bit is GPIOXDETMODE0_12            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX11	[11]	R/W	<p>Specifies Detect mode when GPIOx11 is in Input mode. It is configured the combination of GPIOXDET_EX11(1 bit) + GPIOXDETMODE0_11(2 bit.). Considers GPIOXDET_EX11 as well as GPIOXDETMODE0_11. Note)            - First bit is GPIOXDET_EX11            - Second and third bit is GPIOXDETMODE0_11            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

GPIOXDET_EX10	[10]	R/W	<p>Specifies Detect mode when GPIOx10 is in Input mode. It is configured the combination of GPIOXDET_EX10(1 bit) + GPIOXDETMODE0_10(2 bit.). Considers GPIOXDET_EX10 as well as GPIOXDETMODE0_10. Note)            - First bit is GPIOXDET_EX10            - Second and third bit is GPIOXDETMODE0_10            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX9	[9]	R/W	<p>Specifies Detect mode when GPIOx9 is in Input mode. It is configured the combination of GPIOXDET_EX9(1 bit) + GPIOXDETMODE0_9(2 bit.). Considers GPIOXDET_EX9 as well as GPIOXDETMODE0_9. Note)            - First bit is GPIOXDET_EX9            - Second and third bit is GPIOXDETMODE0_9            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX8	[8]	R/W	<p>Specifies Detect mode when GPIOx8 is in Input mode. It is configured the combination of GPIOXDET_EX8(1 bit) + GPIOXDETMODE0_8(2 bit.). Considers GPIOXDET_EX8 as well as GPIOXDETMODE0_8. Note)            - First bit is GPIOXDET_EX8            - Second and third bit is GPIOXDETMODE0_8            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX7	[7]	R/W	<p>Specifies Detect mode when GPIOx7 is in Input mode. It is configured the combination of GPIOXDET_EX7(1 bit) + GPIOXDETMODE0_7(2 bit.). Considers GPIOXDET_EX7 as well as GPIOXDETMODE0_7. Note)            - First bit is GPIOXDET_EX7            - Second and third bit is GPIOXDETMODE0_7            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX6	[6]	R/W	<p>Specifies Detect mode when GPIOx6 is in Input mode. It is configured the combination of GPIOXDET_EX6(1 bit) + GPIOXDETMODE0_6(2 bit.). Considers GPIOXDET_EX6 as well as GPIOXDETMODE0_6. Note)            - First bit is GPIOXDET_EX6            - Second and third bit is GPIOXDETMODE0_6            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX5	[5]	R/W	<p>Specifies Detect mode when GPIOx5 is in Input mode. It is configured the combination of GPIOXDET_EX5(1 bit) + GPIOXDETMODE0_5(2 bit.). Considers GPIOXDET_EX5 as well as GPIOXDETMODE0_5. Note)            - First bit is GPIOXDET_EX5            - Second and third bit is GPIOXDETMODE0_5            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX4	[4]	R/W	<p>Specifies Detect mode when GPIOx4 is in Input mode. It is configured the combination of GPIOXDET_EX4(1 bit) + GPIOXDETMODE0_4(2 bit.). Considers GPIOXDET_EX4 as well as GPIOXDETMODE0_4. Note)            - First bit is GPIOXDET_EX4            - Second and third bit is GPIOXDETMODE0_4            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

GPIOXDET_EX3	[3]	R/W	<p>Specifies Detect mode when GPIOx3 is in Input mode. It is configured the combination of GPIOXDET_EX3(1 bit) + GPIOXDETMODE0_3(2 bit). Considers GPIOXDET_EX3 as well as GPIOXDETMODE0_3. Note)            - First bit is GPIOXDET_EX3            - Second and third bit is GPIOXDETMODE0_3            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX2	[2]	R/W	<p>Specifies Detect mode when GPIOx2 is in Input mode. It is configured the combination of GPIOXDET_EX2(1 bit) + GPIOXDETMODE0_2(2 bit). Considers GPIOXDET_EX2 as well as GPIOXDETMODE0_2. Note)            - First bit is GPIOXDET_EX2            - Second and third bit is GPIOXDETMODE0_2            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX1	[1]	R/W	<p>Specifies Detect mode when GPIOx1 is in Input mode. It is configured the combination of GPIOXDET_EX1(1 bit) + GPIOXDETMODE0_1(2 bit). Considers GPIOXDET_EX1 as well as GPIOXDETMODE0_1. Note)            - First bit is GPIOXDET_EX1            - Second and third bit is GPIOXDETMODE0_1            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0
GPIOXDET_EX0	[0]	R/W	<p>Specifies Detect mode when GPIOx0 is in Input mode. It is configured the combination of GPIOXDET_EX0(1 bit) + GPIOXDETMODE0_0(2 bit). Considers GPIOXDET_EX0 as well as GPIOXDETMODE0_0. Note)            - First bit is GPIOXDET_EX0            - Second and third bit is GPIOXDETMODE0_0            000 : Low Level 001 : High Level            010 : Falling Edge 011 : Rising Edge            100 : Both(Rising &amp; Falling) Edge 101 111 : Reserved         </p>	0

### 18.3.12 GPIOxDETENB

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x3C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXDETENB	[31:0]	R/W	GPIOx[31:0]: Decides the use of the detected mode of GPIOx PAD. 0: Disable 1: Enable	0

### 18.3.13 GPIOx\_SLEW

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_SLEW	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Slew resistance of GPIOx PAD. 0: Fast Slew 1: Normal Slew	0xFFFFFFFF

### 18.3.14 GPIOx\_SLEW\_DISABLE\_DEFAULT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_SLEW_DISABLE_DEFAULT	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Slew resistance of GPIOx PAD. 0: Use Default Slew 1: Use GPIOx_Slew Value	0xFFFFFFFF

### 18.3.15 GPIOx\_DRV1

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x48 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_DRV1	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Drive Strength1 resistance of GPIOx PAD. 0: Drive Strength0 to 0 1: Drive Strength0 to 1	0

### 18.3.16 GPIOx\_DRV1\_DISABLE\_DEFAULT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x4C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_DRV1_DISABLE_DEFAULT	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Drive Strength1 resistance of GPIOx PAD. 0: Use Default Drive Strength 1: Use GPIOx_DRV1 Value	0xFFFFFFFF

### 18.3.17 GPIOx\_DRV0

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_DRV0	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Drive Strength0 resistance of GPIOx PAD. 0: Drive Strength1 to 0 1: Drive Strength1 to 1	0

### 18.3.18 GPIOx\_DRV0\_DISABLE\_DEFAULT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x54 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_DRV0_DIS- ABLE_DEFAULT	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Drive Strength0 resistance of GPIOx PAD. 0: Use Default Drive Strength 1: Use GPIOx_DRV0 Value	0xFFFFFFFF

### 18.3.19 GPIOx\_PULLSEL

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x58 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_PULLSEL	[31:0]	R/W	GPIOx[31:0]: Decides the Pull-up or Pull-down of GPIOx PAD. 0: Pull-Down 1: Pull-Up	0

### 18.3.20 GPIOx\_PULLSEL\_DISABLE\_DEFAULT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x5C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_PULLSEL_DISABLE_DEFAULT	[31:0]	R/W	GPIOx[31:0]: Decides the use of the PullSel resistance of GPIOx PAD. 0: Use Default Pull Sel 1: Use GPIOx_PULLSEL Value	0

### 18.3.21 GPIOx\_PULLENB

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x60 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_PULLENB	[31:0]	R/W	GPIOx[31:0]: Decides the use of the Pull-up resistance (100 Kohm) of GPIOx PAD. 0: Pull-Up Disable 1: Pull-Up Enable	0

### 18.3.22 GPIOx\_PULLENB\_DISABLE\_DEFAULT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x64 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_PULLENB_DISABLE_DEFAULT	[31:0]	R/W	GPIOx[31:0]: Decides the use of the PullEnb resistance of GPIOx PAD. 0: Use Default Pull Enb 1: Use GPIOx_PULLENB Value	0

### 18.3.23 GPIOx SECURE MARKING REGISTER

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x70 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
SECURE_MARKING_REGISTER	[31:0]	R/W	Specifies the relevant register is secure or non-secure. For example, Secure Marking Register [n] is 0, than GPIOx [n] is non-secure bit, and Secure Marking Register [n] is 1, than GPIOx [n] is secure bit. Only secure CPU can set or clear Secure Marking Register. 0: relevant register is non-secure 1: Use GPIOx_PULLENB Value	0

### 18.3.24 GPIOx\_INPUTENB

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x74 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_INPUTENB	[31:0]	R/W	GPIOx[31:0]: Specifies GPIOx In/Out mode. 0: Output Mode 1: Input Mode	0

### 18.3.25 GPIOx\_INPUTENB\_DISABLE\_DEFAULT

- Base Address: 20700000 (GPIO[0])
- Base Address: 20710000 (GPIO[1])
- Base Address: 20720000 (GPIO[2])
- Base Address: 20730000 (GPIO[3])
- Base Address: 20740000 (GPIO[4])
- Base Address: 20750000 (GPIO[5])
- Base Address: 20760000 (GPIO[6])
- Base Address: 20770000 (GPIO[7])
- Address = Base Address + 0x78 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOX_INPUTENB_DISABLE_DEFAULT	[31:0]	R/W	GPIOx[31:0]: Decides the use of the InputEnb resistance of GPIOx PAD. 0: Use Default Input Enb 1: Use GPIOx_INPUTENB Value	0

# 19 SPI/QSPI

## 19.1 Overview

The SPI/SSP is a full-duplex synchronous serial interface and the QSPI is a half-duplex synchronous serial interface. It supports Serial Peripheral Interface (SPI) and Synchronous Serial Protocol (SSP). It can connect to a variety of external converter, serial memory and many other device which use serial protocols for transferring data. The SPI / SSP standard has four I / O pin signals associated with transmission, and the QSPI has six I / O pin signals associated with transmission: SPI - SSPCLK(SCLK), the SSPRXD(MISO) data receive line, the SSPTXD(MOSI) data transfer line, SSPFSS (Chip Select in SPI mode, Frame Indicator in SSP mode). QSPI - SSPCLK(SCLK), IO\_0, IO\_1, IO\_2, IO\_3, SSPFSS(CS/SS). The DRONE\_SOChas three SPI/SSP port and it can operate in Master and Slave mode

### 19.1.1 Features

- 3-ch SPI, 5-ch QSPI controller
- Supports the Motorola SPI protocol, National Semiconductor Microwire and Texas Instruments SSP
- Supports 8-bit/16-bit/32-bit bus interface
- Master & Slave mode
  - SPI : Master & Slave mode(SPI1/SPI2)
  - QSPI : Master Only
- Supports Dual Data Rate (DDR) - when Dual/Quad mode of operation
- Supports eXecute-In-Place(XIP) - when Dual/Quad mode of operation
- Supports DMA
- FIFO depth - the master has 32 transmit / receive FIFOs, and the slave has 8 transmit / receive FIFOs.
- Programmable delay on the sample time of the received serial data bit (rxn), when configured in Master Mode; enables programmable control of routing delays resulting in higher serial data-bit rates.
- Inform the system that a receive FIFO (over-run/under-run) has occurred
- Inform the system that a multi-master contention has occurred
- Maximum SPI CORE frequency is 200 MHz
  - Master Mode : Max 100MHz
  - Slave Mode : 5MHz (8MHz using DMA)

### 19.1.2 Block Diagram

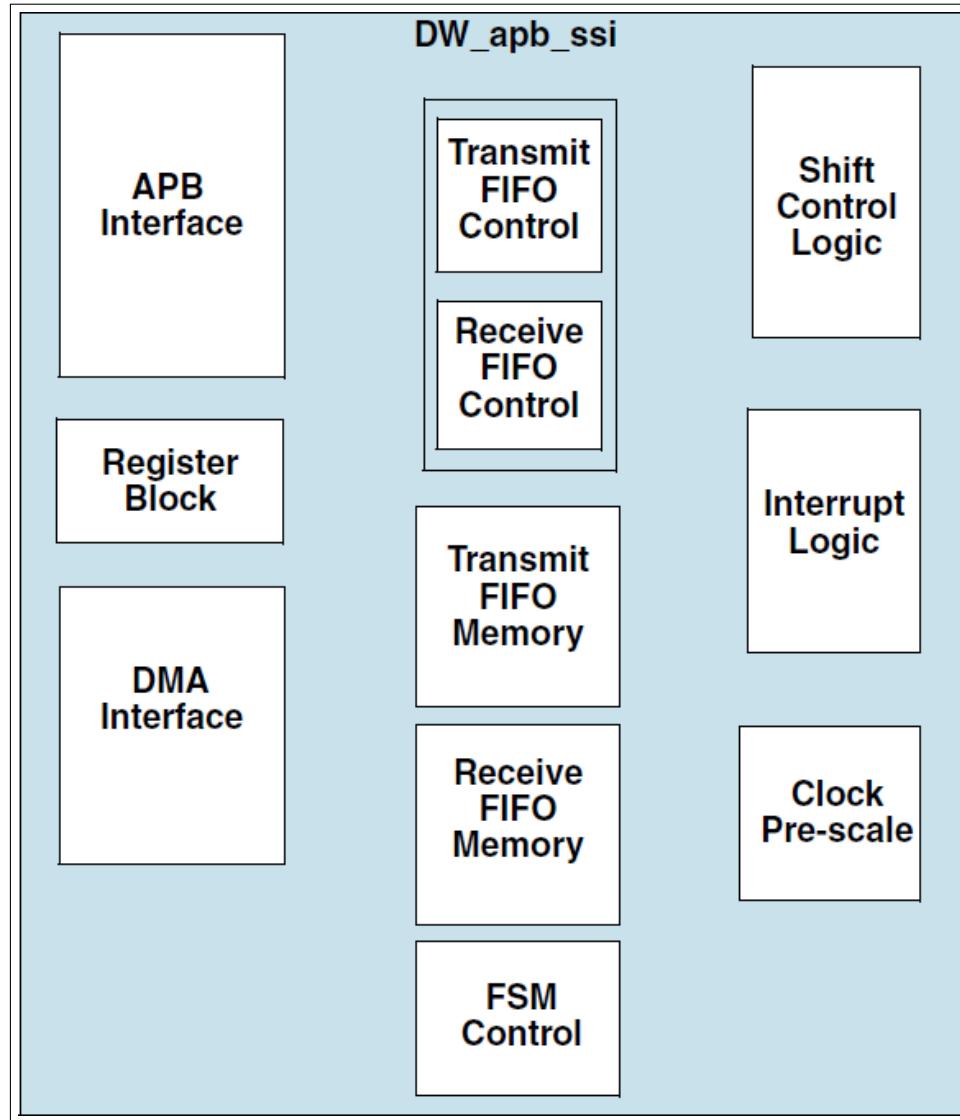


Figure 19.1: SPI Block Diagram

## 19.2 Functional Description

The DRONE\_SOC SPI is a configurable, synthesizable, and programmable component that is a full-duplex master or slave-synchronous serial interface. The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI may also interface with a DMA Controller using an optional set of DMA signals, which can be selected at configuration time.

As described in more detail later, the SPI can be configured in one of two modes of operations: as a serial master or a serial slave. The SPI can connect to any serial-master or serial-slave peripheral device using one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI)
- Texas Instruments Serial Protocol (SSP)
- National Semiconductor Microwire

### 19.2.1 Reset Configuration

The SPI/SSP is reset by the global reset signal PRESETn. In DRONE\_SOC, PRESETn is controlled by CMU. User can set up SPI/SSP reset signals by CPU.

### 19.2.2 Clock Configuration

CMU has various clocks, such as spi\_0\_core\_clk, spi\_0\_apb\_clk, spi\_1\_core\_clk, spi\_1\_apb\_clk, etc. When clock name is spi\_#\_core\_clk, # means module index of SPI. So, spi\_0\_core\_clk and spi\_0\_apb\_clk clocks are used for SPI[0], spi\_1\_core\_clk and spi\_1\_apb\_clk clocks are used for SPI[1] and spi\_2\_core\_clk and spi\_2\_apb\_clk clocks are used for SPI[2]. From now on, just use spi\_0\_core\_clk and spi\_0\_apb\_clk clock name. But user must setting exact clock for each SPI module index.

The spi\_0\_core\_clk is used when the SPI/SSP transmit or receive SPI/SSP Protocol. spi\_0\_core\_clk is generated by Clock Generator of the SPI/SSP. Each SPI/SSP has own Clock Divider in CMU. Therefore, users must set up the SPI/SSP Clock Generator before the SPI configuration stage.

SPI/SSP provides a variety of clocks. As described in the upper figure, the SPI uses spi\_0\_core\_clk, which is from CMU. You can also select spi\_0\_core\_clk from various clock sources.

SPI has an internal 2x clock divider. SCLK\_SPI should be configured to have a double of the SPI operating clock frequency.

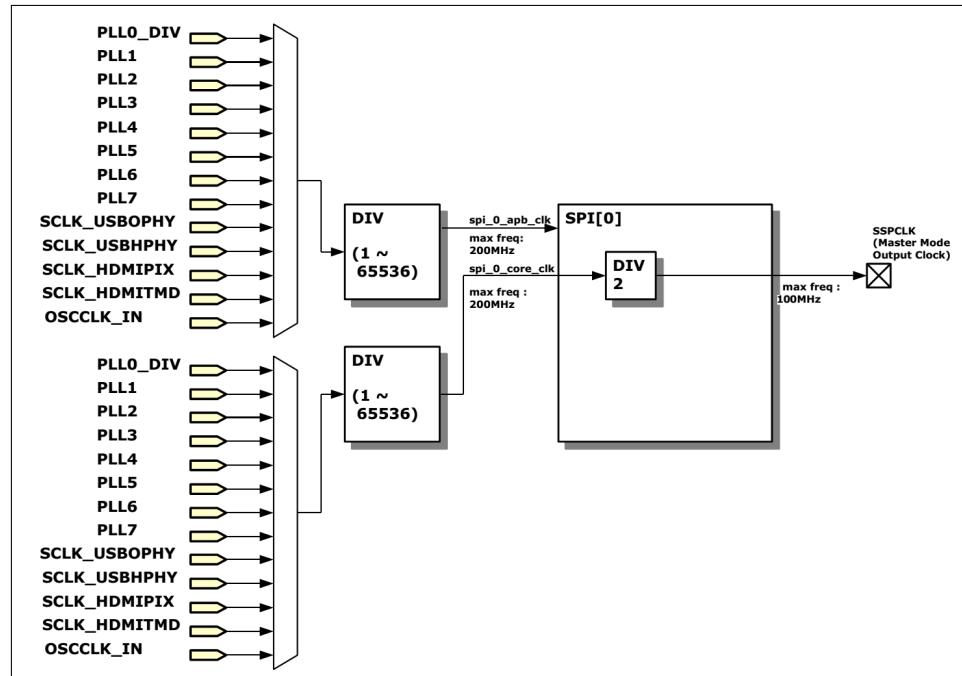


Figure 19.2: SPI Clock Diagram

### 19.2.3 Clock Ratios

When DW\_apb\_ssi is configured as a master device, the maximum frequency of the bit-rate clock (sclk\_out) is one-half the frequency of ssi\_clk. This allows the shift control logic to capture data on one clock edge of sclk\_out and propagate data on the opposite edge.

Figure 19.3 on page 730 illustrates the maximum ratio between sclk\_out and ssi\_clk.

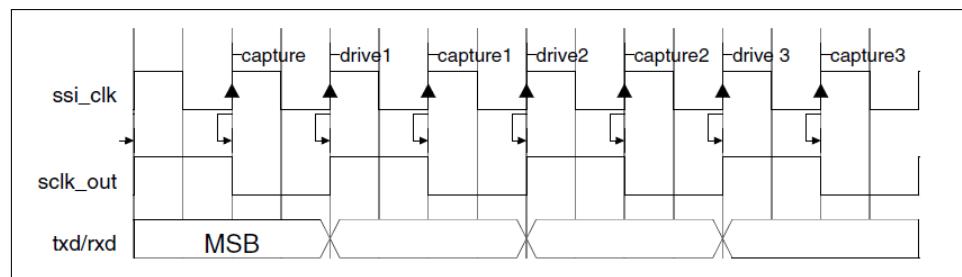


Figure 19.3: Maximum sclk\_out/ssi\_clk Ratio

The sclk\_out line toggles only when an active transfer is in progress. At all other times it is held in an inactive state, as defined by the serial protocol under which it operates.

The frequency of sclk\_out can be derived from the following equation:

$$F_{sclk\_out} = \frac{F_{ssi\_clk}}{SCKDV}$$

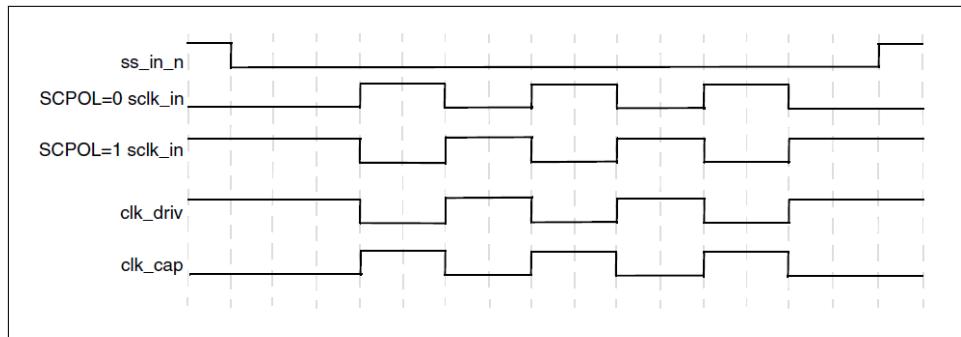
SCKDV is a bit field in the programmable register BAUDR, holding any even value in the range 0 to 65,534. If SCKDV is 0, then sclk\_out is disabled.

When DW\_apb\_ssi is configured as a slave device, the minimum frequency of ssi\_clk depends on the configuration parameter and the operation of the slave peripheral.

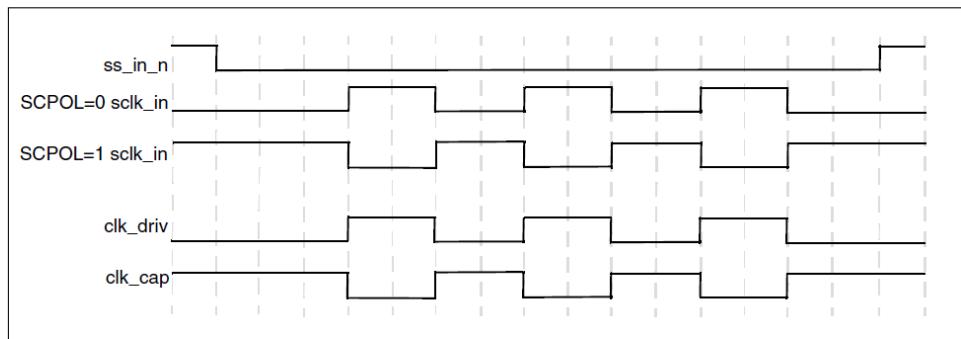
the transmit and receive shift registers work directly with the bit-rate clock from the master device (sclk\_in) to eliminate the need for synchronization.

To reduce the synchronization delay, the synchronization scheme uses two flip flops: one works on the positive edge of ssi\_clk; and other works on the negative edge of ssi\_clk. These flip flops reduce the synchronization delay to one ssi\_clk cycle and enable SPI to work on lower clock ratios. The SPI slave device minimum frequency of ssi\_clk is 4 times the maximum expected frequency of the bit-rate clock (sclk\_in).

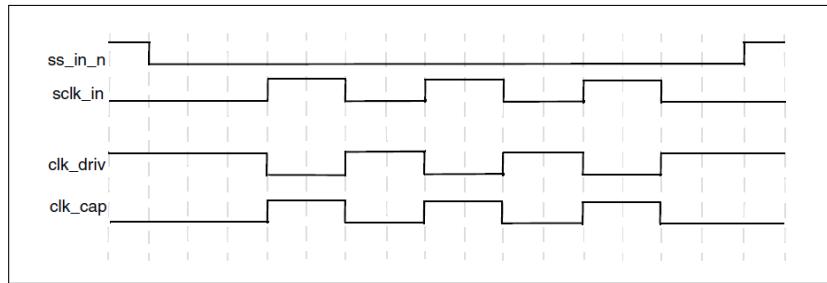
The bit-rate clock is gated according to different capture and driving edges for different frame formats. The capture clock (clk\_cap) is used in the receive shifter to capture the data on the rxd line. The driving clock (clk\_driv) is used in the transmit shifter to drive the data on the txd line. Figure 19.4, Figure 19.5, Figure 19.6 and Figure 19.7 illustrate how capture and driving clocks are derived from bit-rate clock for different frame formats.



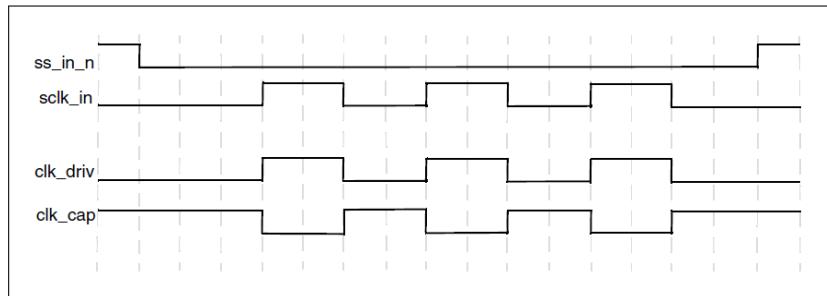
**Figure 19.4:** Frame Format for Motorola Serial Peripheral Interface (SPI) with SCPH = 0



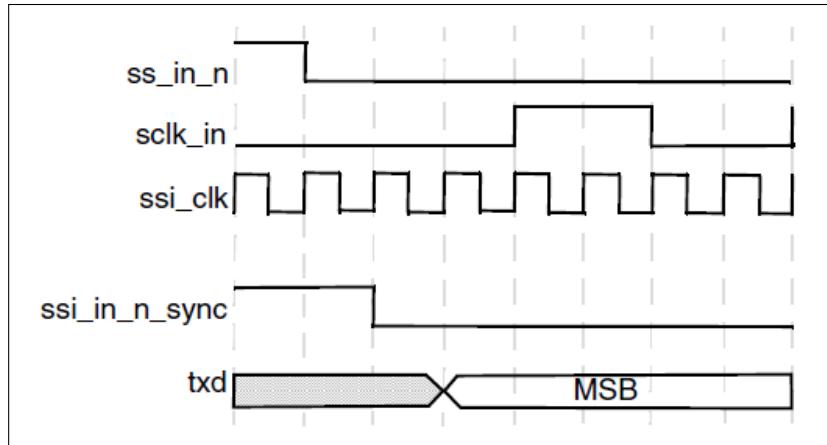
**Figure 19.5:** Frame Format for Motorola Serial Peripheral Interface (SPI) with SCPH = 1



**Figure 19.6:** National Semiconductors Microwire Frame Format



**Figure 19.7:** Texas Instruments Synchronous Serial Protocol (SSP)



**Figure 19.8:** Driving TXD When SCPH = 0

If the frame format is programmed for SPI with `SCPH=0`, then before the first edge of bit-rate clock (`sclk_in`) arrives, the data must be present on the `txd` line. Internally to drive data on '`txd`' line the synchronized version of the slave select (`ss_in_n`) signal is used. This mechanism takes 2 cycles on `ssi_clk` to complete and the data is driven on the 3rd cycle of `ssi_clk` as shown in Figure 19.8. Therefore, to capture the data correctly, the first edge of bit-rate clock (`sclk_in`) must arrive after at least 4 `ssi_clk` cycles from when the slave is selected (`ss_in_n`).

### 19.2.3.1 Frequency Ratio Summary

A summary of the frequency ratio restrictions between the bit-rate clock (`sclk_out/sclk_in`) and the `DW_apb_ssi` peripheral clock (`ssi_clk`) are as follows:

- Master

$$F_{ssi\_clk} \geq 2 \times (\text{maximum } F_{sclk\_out})$$

- Slave

$$F_{ssi\_clk} \geq 4 \times (\text{maximum } F_{sclk\_in})$$

## 19.2.4 Transfer Modes

When transferring data on the serial bus, the DW\_apb\_ssi operates in the modes discussed in this section. The transfer mode (TMOD) is set by writing to control register 0 (CTRLR0), as described in “Register Descriptions” on page 767.

### 19.2.4.1 Transmit and Receive

When TMOD = 2'b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

### 19.2.4.2 Transmit Only

When TMOD = 2'b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

### 19.2.4.3 Receive Only

When TMOD = 2'b10, the transmit data are invalid. When configured as a slave, the transmit FIFO is never popped in Receive Only mode. The txd output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

### 19.2.4.4 EEPROM Read

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as the DW\_apb\_ssi master is transmitting data on its txd line, data on the rxd line is ignored). The DW\_apb\_ssi master continues to transmit data until the transmit FIFO is empty. Therefore, you should ONLY have enough data frames in the transmit FIFO to supply the opcode and address to the EEPROM. If more data frames are in the transmit FIFO than are needed, then read data is lost.

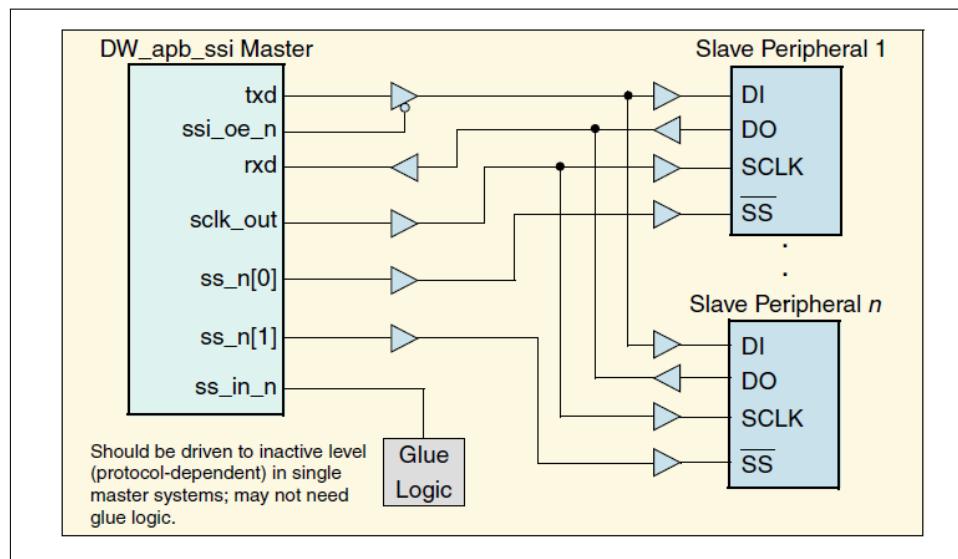
When the transmit FIFO becomes empty (all control information has been sent), data on the receive line (rxd) is valid and is stored in the receive FIFO; the txd output is held at a constant logic level. The serial transfer continues until the number of data frames received by the DW\_apb\_ssi master matches the value of the NDF field in the CTRLR1 register + 1.

### 19.2.5 Operation Modes

The DW\_apb\_ssi can be configured in the fundamental modes of operation discussed in this section.

#### 19.2.5.1 Serial Master Mode

This mode enables serial communication with serial-slave peripheral devices. When configured as a serial-master device, the DW\_apb\_ssi initiates and controls all serial transfers. Figure 19.9 shows an example of the DW\_apb\_ssi configured as a serial master with all other devices on the serial bus configured as serial slaves.



**Figure 19.9:** DW\_apb\_ssi Configured as Master Device

The serial bit-rate clock, generated and controlled by the DW\_apb\_ssi, is driven out on the sclk\_out line. When the DW\_apb\_ssi is disabled (`SSI_EN = 0`), no serial transfers can occur and `sclk_out` is held in “inactive” state, as defined by the serial protocol under which it operates.

**19.2.5.1.1 Data Transfers** Data transfers are started by the serial-master device. When the DW\_apb\_ssi is enabled (`SSI_EN=1`), at least one valid data entry is present in the transmit FIFO and a serial-slave device is selected. When actively transferring data, the busy flag (`BUSY`) in the status register (`SR`) is set. You must wait until the busy flag is cleared before attempting a new serial transfer.

**19.2.5.1.2 Master SPI and SSP Serial Transfers** The sections “Motorola Serial Peripheral Interface (SPI)” on page 742 and “Texas Instruments Synchronous Serial Protocol (SSP)” on page 747 describe the SPI and SSP serial protocols, respectively. They include timing diagrams and provide information as to how data are structured in the transmit and receive the FIFOs before and after the serial transfer. When the transfer mode is “ransmit and receive”

or “transmit only” (TMOD = 2'b00 or TMOD = 2'b01, respectively), transfers are terminated by the shift control logic when the transmit FIFO is empty. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level (TXFTLR) can be used to early interrupt (ssi\_txe\_intr) the processor indicating that the transmit FIFO buffer is nearly empty. When a DMA is used for APB accesses, the transmit data level (DMATDLR) can be used to early request (dma\_tx\_req) the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer. The user may also write a block of data (at least two FIFO entries) into the transmit FIFO before enabling a serial slave. This ensures that serial transmission does not begin until the number of data-frames that make up the continuous transfer are present in the transmit FIFO. When the transfer mode is “receive only”(TMOD = 2'b10), a serial transfer is started by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. The txd output from the SPI is held at a constant logic level for the duration of the serial transfer. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the “umber of data frames” (NDF) field in control register 1 (CTRLR1). If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23; the receive logic terminates the serial transfer when the number of frames received is equal to the NDF value + 1. This transfer mode increases the bandwidth of the APB bus as the transmit FIFO never needs to be serviced during the transfer. The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. When the transfer mode is “eprom\_read” (TMOD = 2'b11), a serial transfer is started by writing the opcode and/or address into the transmit FIFO when a serial slave (EEPROM) is selected. The opcode and address are transmitted to the EEPROM device, after which read data is received from the EEPROM device and stored in the receive FIFO. The end of the serial transfer is controlled by the NDF field in the control register 1 (CTRLR1).

The receive FIFO threshold level (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA is used for APB accesses, the receive data level (DMARDLR) can be used to early request (dma\_rx\_req) the DMA Controller, indicating that the receive FIFO is nearly full. A typical software flow for completing an SPI or SSP serial transfer from the SPI serial master is outlined as follows:

1. If the DW\_apb\_ssi is enabled, disable it by writing 0 to the SSI Enable register (SSIENR).
2. Set up the DW\_apb\_ssi control registers for the transfer; these registers can be set in any order.
  - Write Control Register 0 (CTRLR0). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
  - If the transfer mode is receive only, write CTRLR1 (Control Register 1) with the number of frames in the transfer minus 1; for example, if you want to receive four data frames, write this register with 3.
  - Write the Baud Rate Select Register (BAUDR) to set the baud rate for the transfer.
  - Write the Transmit and Receive FIFO Threshold Level registers (TXFTLR and RXFTLR, respectively) to set FIFO threshold levels.
  - Write the IMR register to set up interrupt masks.
  - The Slave Enable Register (SER) register can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data Register (DR), the transfer does not begin until a slave is enabled.
3. Enable the DW\_apb\_ssi by writing 1 to the SSIENR register.
4. Write data for transmission to the target slave into the transmit FIFO (write DR). If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately; for more information, see the note on page 734. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).

6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is receive only ( $\text{TMOD} = 2\text{b}10$ ), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0.
7. If the transfer mode is not transmit only ( $\text{TMOD} \neq 01$ ), read the receive FIFO until it is empty.
8. Disable the DW\_apb\_ssi by writing 0 to SSIENR.

Figure 19.10 shows a typical software flow for starting a DW\_apb\_ssi master SPI/SSP serial transfer. The diagram also shows the hardware flow inside the serial-master component.

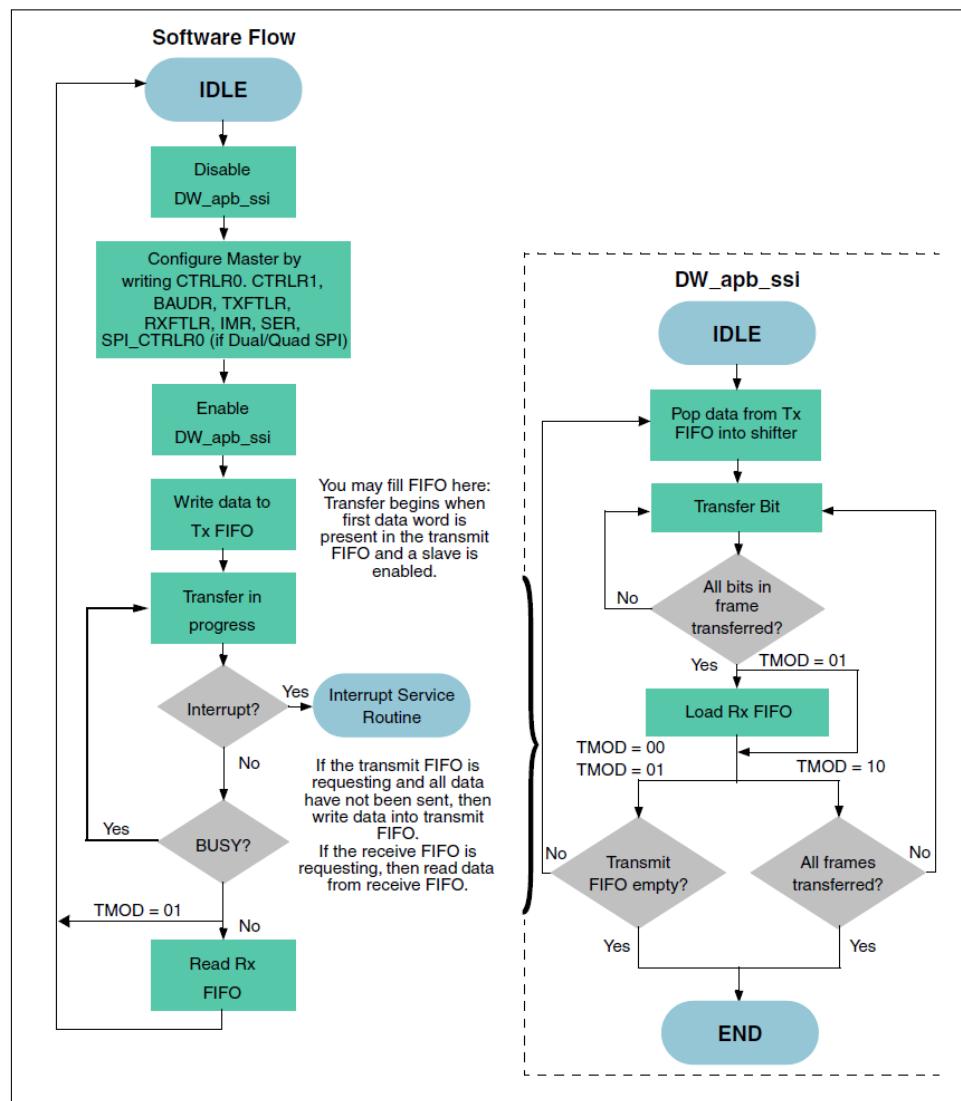


Figure 19.10: DW\_apb\_ssi Master SPI/SSP Transfer Flow

**19.2.5.1.3 Master Microwire Serial Transfers** Microwire serial transfers from the DW\_apb\_ssi serial master are controlled by the Microwire Control Register (MWCR). The MWHS bit field enables and disables the Microwire handshaking interface. The MDD bit field controls the direction of the data frame (the control frame is always transmitted by the master and received by the slave). The MWMOD bit field defines whether the transfer is sequential or nonsequential.

All Microwire transfers are started by the DW\_apb\_ssi serial master when there is at least one control word in the transmit FIFO and a slave is enabled. When the DW\_apb\_ssi master transmits the data frame (MDD = 1), the transfer is terminated by the shift logic when the transmit FIFO is empty. When the DW\_apb\_ssi master receives the data frame (MDD = 1), the termination of the transfer depends on the setting of the MWMOD bit field. If the transfer is nonsequential (MWMOD = 0), it is terminated when the transmit FIFO is empty after shifting in the data frame from the slave. When the transfer is sequential (MWMOD = 1), it is terminated by the shift logic when the number of data frames received is equal to the value in the CTRLR1 register + 1.

When the handshaking interface on the DW\_apb\_ssi master is enabled (MWHS = 1), the status of the target slave is polled after transmission. Only when the slave reports a ready status does the DW\_apb\_ssi master complete the transfer and clear its BUSY status. If the transfer is continuous, the next control/data frame is not sent until the slave device returns a ready status.

A typical software flow for completing a Microwire serial transfer from the DW\_apb\_ssi serial master is outlined as follows:

1. If the DW\_apb\_ssi is enabled, disable it by writing 0 to SSIENR.
2. Set up the DW\_apb\_ssi control registers for the transfer. These registers can be set in any order. Write CTRLR0 to set transfer parameters.
  - If the transfer is sequential and the DW\_apb\_ssi master receives data, write CTRLR1 with the number of frames in the transfer minus 1; for instance, if you want to receive four data frames, write this register with 3.
  - Write BAUDR to set the baud rate for the transfer.
  - Write TXFTLR and RXFTLR to set FIFO threshold levels.
  - Write the IMR register to set up interrupt masks. You can write the SER register to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the DR register, the transfer does not begin until a slave is enabled.
3. Enable the DW\_apb\_ssi by writing 1 to the SSIENR register.
4. If the DW\_apb\_ssi master transmits data, write the control and data words into the transmit FIFO (write DR). If the DW\_apb\_ssi master receives data, write the control word(s) into the transmit FIFO. If no slaves were enabled in the SER register at this point, enable now to begin the transfer.
5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately; for more information, see the note on page 734. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is sequential and the DW\_apb\_ssi master receives data, the transfer is stopped by the shift control logic when the specified number of data frames is received. When the transfer is done, the BUSY status is reset to 0.
7. If the DW\_apb\_ssi master receives data, read the receive FIFO until it is empty.
8. Disable the DW\_apb\_ssi by writing 0 to SSIENR.

Figure 19.11 shows a typical software flow for starting a DW\_apb\_ssi master Microwire serial transfer. The diagram also shows the hardware flow inside the serial-master component.

### 19.2.5.2 Serial-Slave Mode

This mode enables serial communication with master peripheral devices. When the DW\_apb\_ssi is configured as a slave device, all serial transfers are initiated and controlled by the serial bus master. Figure 19.12 shows an example

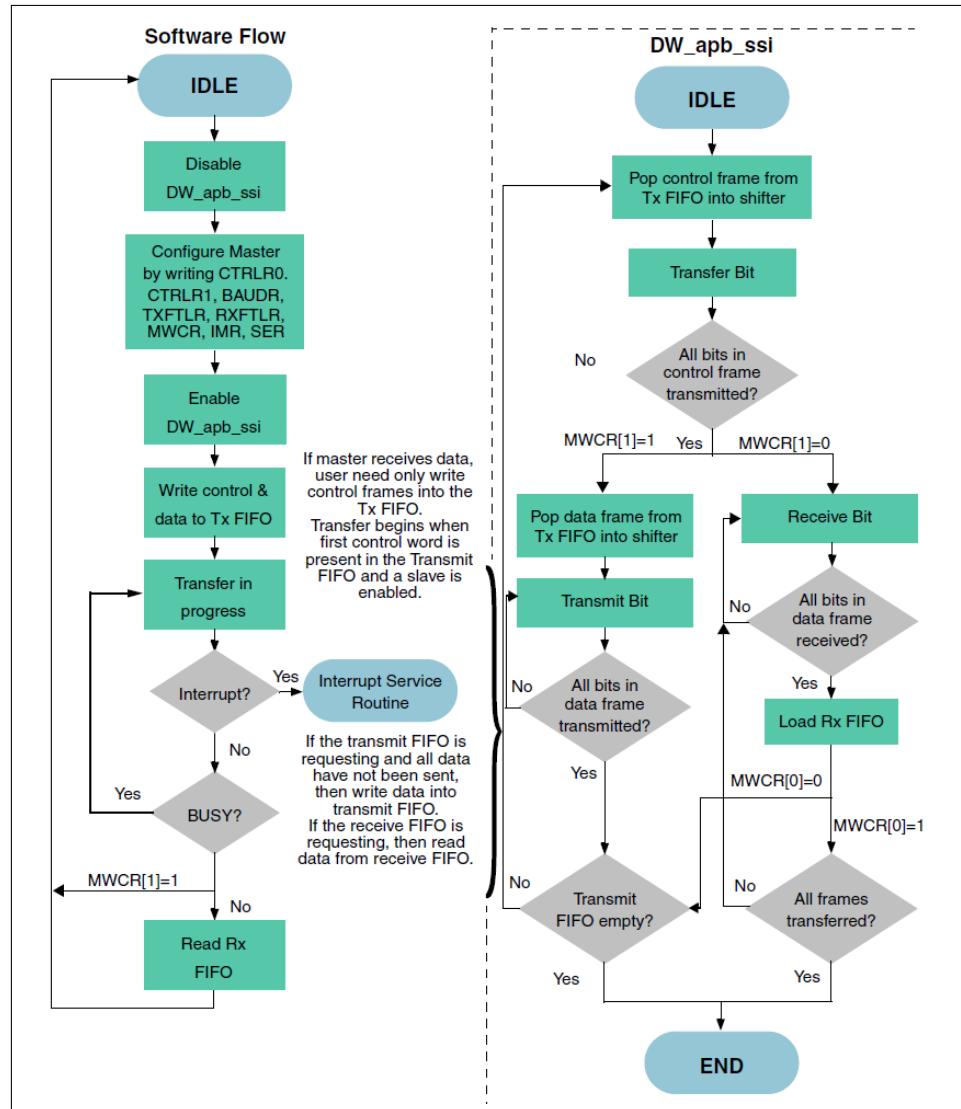
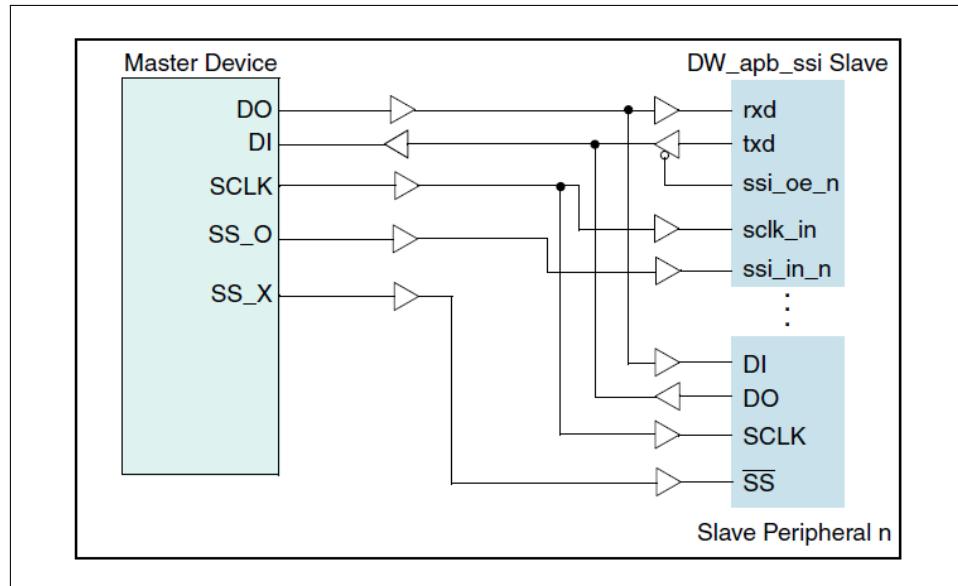


Figure 19.11: DW\_apb\_ssi Master Microwire Transfer Flow

of the DW\_apb\_ssi configured as a serial slave in a single-master bus system. When the DW\_apb\_ssi serial slave is selected during configuration, it enables its txd data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line (sclk\_in), driven from the serial-master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

When the DW\_apb\_ssi serial slave is not selected, it must not interfere with data transfers between the serial-master and other serial-slave devices. When the serial slave is not selected, its txd output is buffered, resulting in a high impedance drive onto the serial master rxd line. The buffers shown in Figure 19.12 are external to DW\_apb\_ssi. The serial clock that regulates the data transfer is generated by the serial-master device and input to the DW\_apb\_ssi slave on sclk\_in. The slave remains in an idle state until selected by the bus master. When not actively transmitting data, the slave must hold its txd line in a high impedance state to avoid interference with serial transfers to other slave devices. The ssi\_oe\_n line is available for use to control the txd output buffer. The slave continues to transfer data to and from the master device as long as it is selected. If the master transmits to all serial slaves, a control bit (SLV\_OE) in the DW\_apb\_ssi control register 0 (CTRLR0) can be programmed to inform the slave if it should respond with data from its txd line.



**Figure 19.12:** DW\_apb\_ssi Configured as Slave Device

**19.2.5.2.1 Slave SPI and SSP Serial Transfers** “Motorola Serial Peripheral Interface (SPI)” on page 742 and “Texas Instruments Synchronous Serial Protocol (SSP)” on page 747 contain a description of the SPI and SSP serial protocols, respectively. The sections also provide timing diagrams and information on how data are structured in the transmit and receive FIFOs before and after the serial transfer.

If the DW\_apb\_ssi slave is receive only ( $\text{TMOD}=2'b10$ ), the transmit FIFO need not contain valid data because the data currently in the transmit shift register is resent each time the slave device is selected. The TXE error flag in the status register (SR) is not set when  $\text{TMOD}=2'b01$ . You should mask the transmit FIFO empty interrupt when this mode is used.

If the DW\_apb\_ssi slave transmits data to the master, you must ensure that data exists in the transmit FIFO before a transfer is initiated by the serial-master device. If the master initiates a transfer to the DW\_apb\_ssi slave when no data exists in the transmit FIFO, an error flag (TXE) is set in the DW\_apb\_ssi status register, and the previously transmitted data frame is resent on txd. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level register (TXFTLR) can be used to early interrupt (ssi\_txe\_intr) the processor, indicating that the transmit FIFO buffer is nearly empty. When a DMA Controller is used for APB accesses, the DMA transmit data level register (DMATDLR) can be used to early request (dma\_tx\_req) the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer. The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level register (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA Controller is used for APB accesses, the DMA receive data level register (DMARDLR) can be used to early request (dma\_rx\_req) the DMA controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing a continuous serial transfer from a serial master to the DW\_apb\_ssi slave is described as follows:

1. If the DW\_apb\_ssi is enabled, disable it by writing 0 to SSIENR.
2. Set up the DW\_apb\_ssi control registers for the transfer. These registers can be set in any order.
  - a. Write CTRLR0 (for SPI transfers SCPH and SCPOL must be set identical to the master device).

- b. Write TXFTLR and RXFTLR to set FIFO threshold levels.
  - c. Write the IMR register to set up interrupt masks.
3. Enable the DW\_apb\_ssi by writing 1 to the SSIENR register.
  4. If the transfer mode is transmit and receive (TMOD=2'b00) or transmit only (TMOD=2'b01), write data for transmission to the master into the transmit FIFO (Write DR). If the transfer mode is receive only (TMOD=2'b10), there is no need to write data into the transmit FIFO; the current value in the transmit shift register is retransmitted.
  5. The DW\_apb\_ssi slave is now ready for the serial transfer. The transfer begins when the DW\_apb\_ssi slave is selected by a serial-master device.
  6. When the transfer is underway, the BUSY status can be polled to return the transfer status. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
  7. The transfer ends when the serial master removes the select input to the DW\_apb\_ssi slave. When the transfer is completed, the BUSY status is reset to 0.
  8. If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO until empty.
  9. Disable the DW\_apb\_ssi by writing 0 to SSIENR.

Figure 19.13 shows a typical software flow for a DW\_apb\_ssi slave SPI or SSP serial transfer. The diagram also shows the hardware flow inside the serial-slave component.

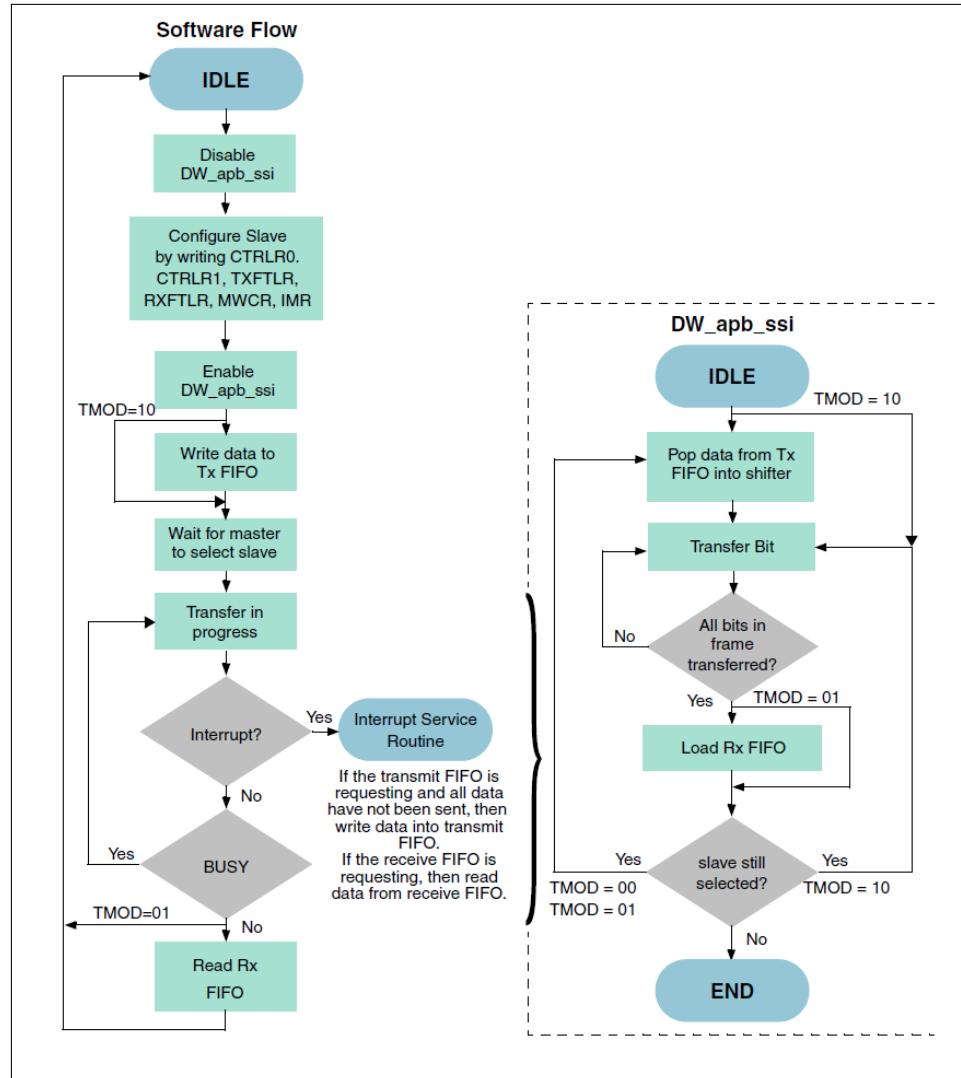


Figure 19.13: DW\_apb\_ssi Slave SPI/SSP Transfer Flow

**19.2.5.2.2 Slave Microwire Serial Transfers** When the DW\_apb\_ssi is configured as a slave device, the Microwire protocol operates in much the same way as the SPI protocol. There is no decode of the control frame by the DW\_apb\_ssi slave device.

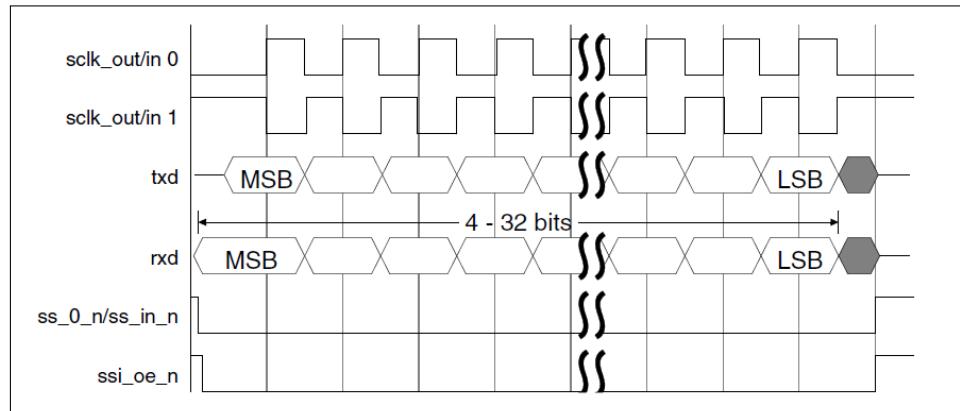
## 19.2.6 Partner Connection Interfaces

The DW\_apb\_ssi can connect to any serial-master or serial-slave peripheral device using one of the interfaces discussed in the following sections.

### 19.2.6.1 Motorola Serial Peripheral Interface (SPI)

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 16/32-bits (depending upon SSI\_MAX\_XFER\_SIZE) in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. Figure 19.14 shows a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPOL = 1.



**Figure 19.14:** SPI Serial Format (SCPH = 0)

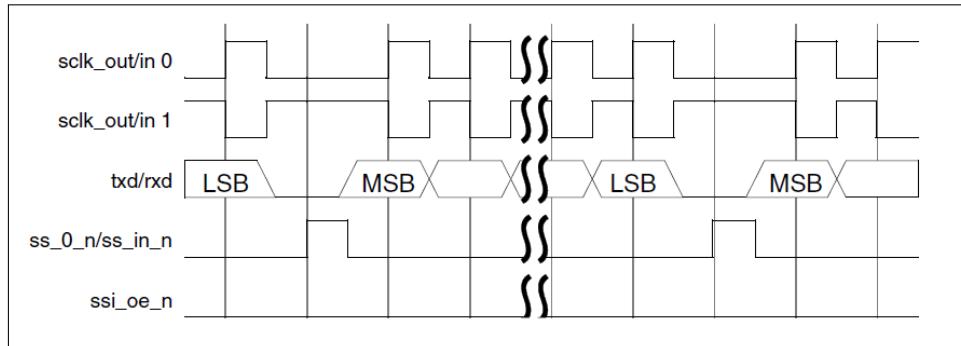
The following signals are illustrated in the timing diagrams in this section:

- sclk\_out serial clock from DW\_apb\_ssi master (master configuration only)
- sclk\_in serial clock from DW\_apb\_ssi slave (slave configuration only)
- ss\_0\_n slave select signal from DW\_apb\_ssi master (master configuration only)
- ss\_in\_n slave select input to the DW\_apb\_ssi slave
- ss\_oe\_n output enable for the DW\_apb\_ssi master/slave
- txd transmit data line for the DW\_apb\_ssi master/slave
- rxd receive data line for the DW\_apb\_ssi master/slave

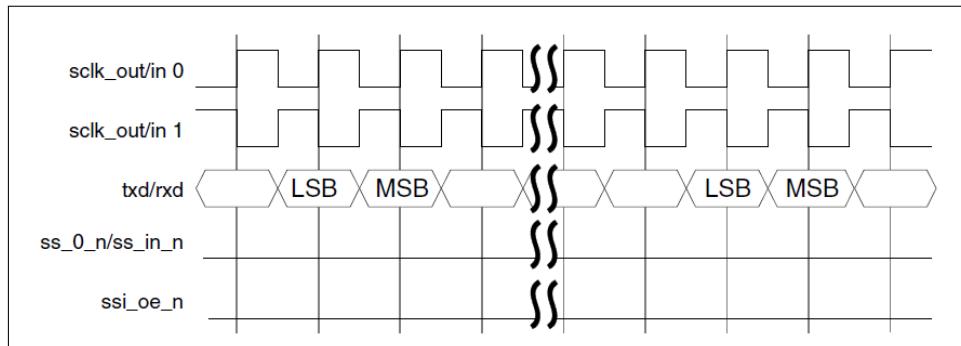
Two different modes of continuous data transfers are supported when SCPH = 0; the selection of the desired operation mode is done using the configuration of the SSI\_SCPH0\_SSTOGGLE parameter:

- When SSI\_SCPH0\_SSTOGGLE is configured as True and CTRLR0. SSTE is set to 1, the DW\_apb\_ssi toggles the slave select signal between frames and the serial clock is held to its default value while the slave select signal is active; this operating mode is illustrated in Figure 19.15.

- When SSI\_SCPH0\_SSTOGGLE is configured as True and CTRLR0.SSTE is set to 0 or when SSI\_SCPH0\_SSTOGGLE is configured as False, the slave select signal stays low and the serial clock runs continuously for the duration of the transfer; this operating mode is illustrated in Figure 19.16.



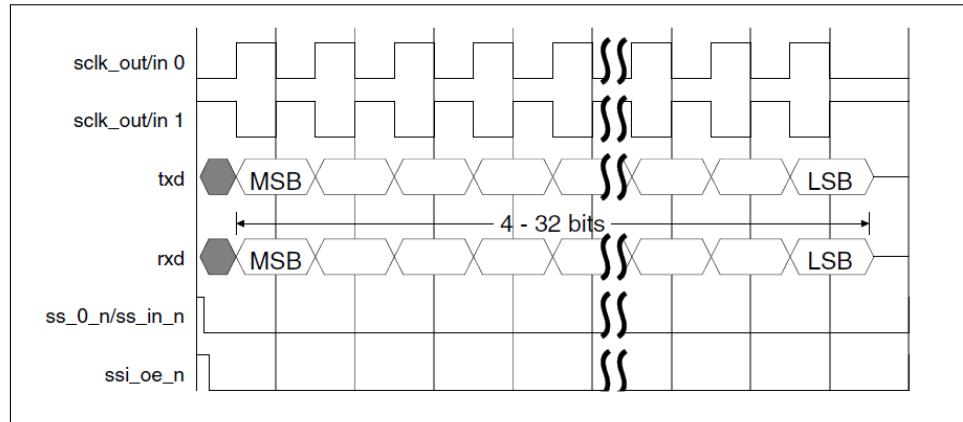
**Figure 19.15:** Serial Format Continuous Transfers (SCPH = 0) when SSI\_SCPH0\_SSTOGGLE = 1



**Figure 19.16:** Serial Format Continuous Transfers (SCPH=0) when SSI\_SCPH0\_SSTOGGLE = 0

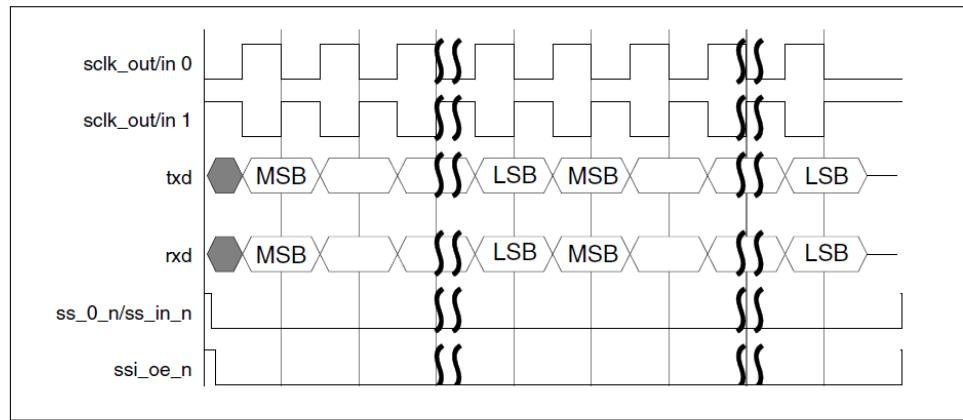
When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured.

Figure 19.17 shows the timing diagram for the SPI format when the configuration parameter SCPH = 1.



**Figure 19.17:** SPI Serial Format (SCPH = 1)

Continuous data frames are transferred in the same way as single frames, with the MSB of the next frame following directly after the LSB of the current frame. The slave select signal is held active for the duration of the transfer. Figure 19.18 shows the timing diagram for continuous SPI transfers when the configuration parameter SCPH = 1.



**Figure 19.18:** SPI Serial Format Continuous Transfer (SCPH = 1)

There are four possible transfer modes on the DW\_apb\_ssi for performing SPI serial transactions; refer to “Transfer Modes” on page 733. For transmit and receive transfers (transfer mode field (9:8) of the Control Register 0 = 2'b00), data transmitted from the DW\_apb\_ssi to the external serial device is written into the transmit FIFO. Data received from the external serial device into the DW\_apb\_ssi is pushed into the receive FIFO.

Figure 19.19 shows the FIFO levels prior to the beginning of a serial transfer and the FIFO levels on completion of the transfer. In this example, two data words are transmitted from the DW\_apb\_ssi to the external serial device in a continuous transfer. The external serial device also responds with two data words for the DW\_apb\_ssi.

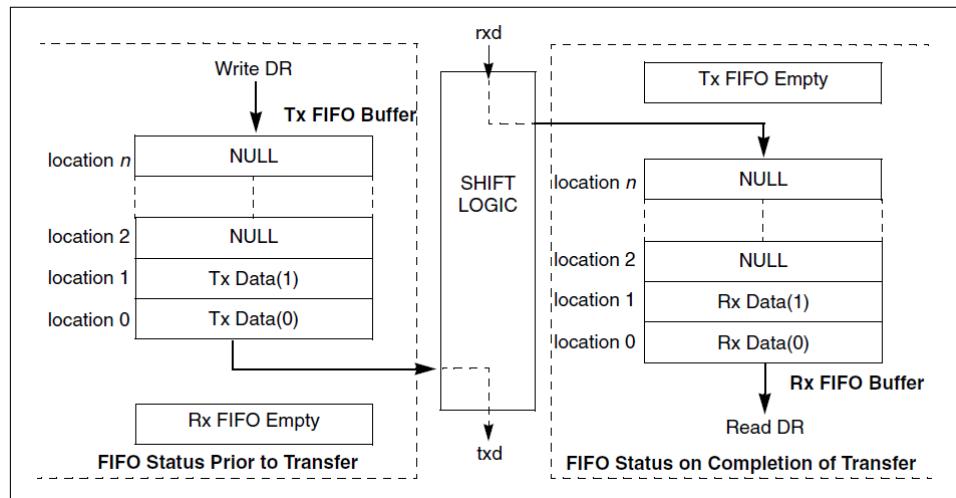


Figure 19.19: FIFO Status for Transmit/Receive SPI and SSP Transfers

For transmit only transfers (transfer mode field (9:8) of the Control Register 0 = 2'b01), data transmitted from the DW\_apb\_ssi to the external serial device is written into the transmit FIFO. As the data received from the external serial device is deemed invalid, it is not stored in the DW\_apb\_ssi receive FIFO.

Figure 19.20 on page 745 shows the FIFO levels prior to the beginning of a serial transfer and the FIFO levels on completion of the transfer. In this example, two data words are transmitted from the DW\_apb\_ssi to the external serial device in a continuous transfer.

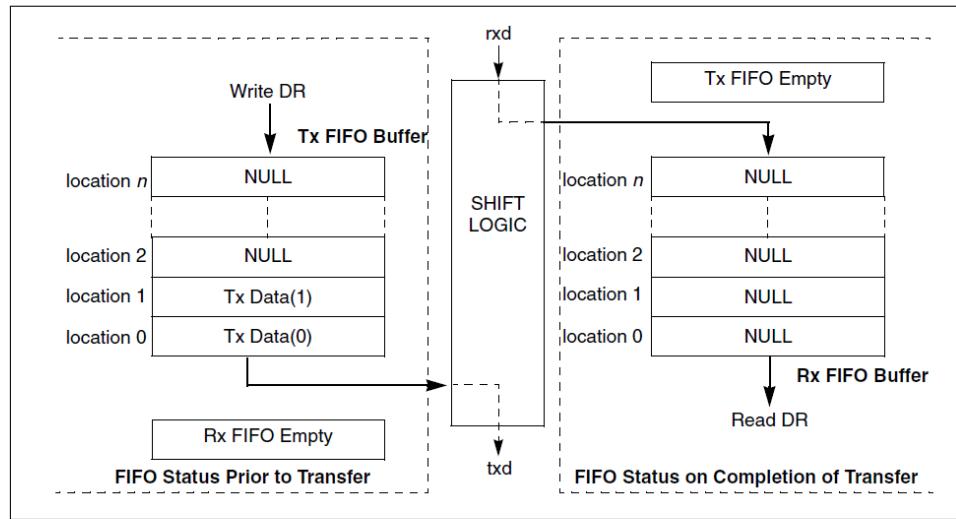
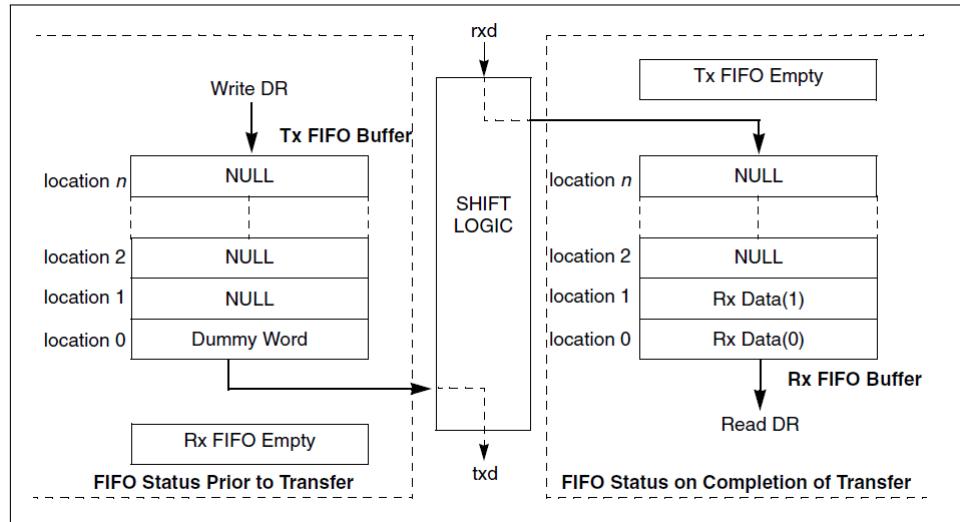


Figure 19.20: FIFO Status for Transmit Only SPI and SSP Transfers

For receive only transfers (transfer mode field (9:8) of the Control Register 0 = 2'b10), data transmitted from the DW\_apb\_ssi to the external serial device is invalid, so a single dummy word is written into the transmit FIFO to begin the serial transfer. The txd output from the DW\_apb\_ssi is held at a constant logic level for the duration of the serial transfer. Data received from the external serial device into the DW\_apb\_ssi is pushed into the receive FIFO.

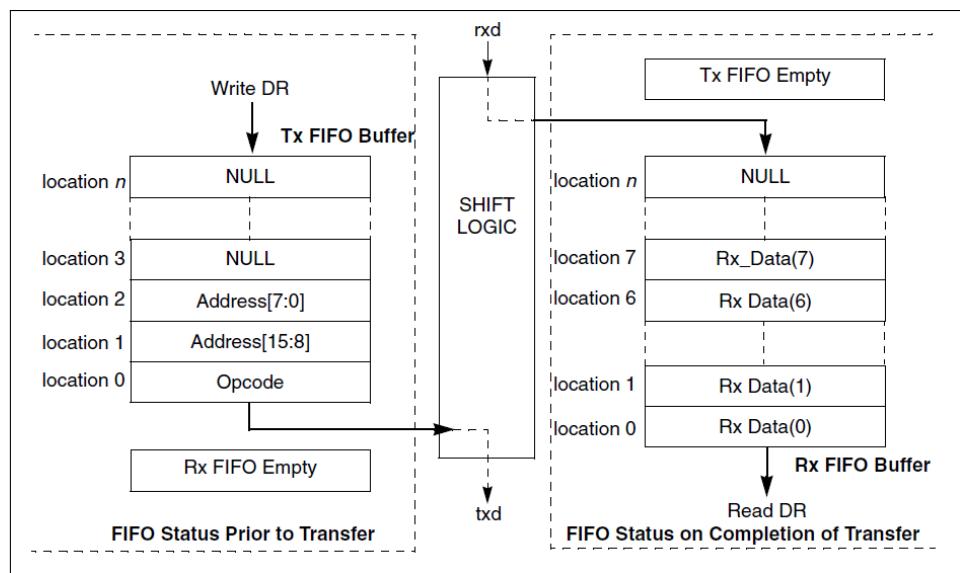
Figure 19.21 on page 746 shows the FIFO levels prior to the beginning of a serial transfer and the FIFO levels on completion of the transfer. In this example, two data words are received by the DW\_apb\_ssi from the external serial device in a continuous transfer.



**Figure 19.21:** FIFO Status for Receive Only SPI and SSP Transfers

For eeprom\_read transfers (transfer mode field [9:8] of the Control Register 0 = 2'b11), opcode and/or EEPROM address are written into the transmit FIFO. During transmission of these control frames, received data is not captured by the DW\_apb\_ssi master. After the control frames have been transmitted, receive data from the EEPROM is stored in the receive FIFO.

Figure 19.22 on page 746 shows the FIFO levels prior to the beginning of a serial transfer and the FIFO levels on completion of the transfer. In this example, one opcode and an upper and lower address are transmitted to the EEPROM, and eight data frames are read from the EEPROM and stored in the receive FIFO of the DW\_apb\_ssi master.

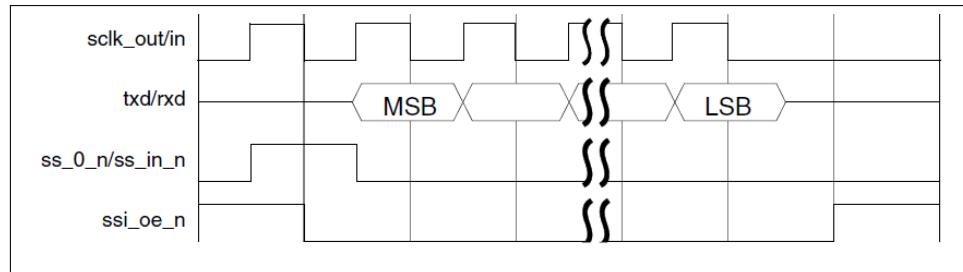


**Figure 19.22:** FIFO Status for EEPROM Read Transfer Mode

### 19.2.6.2 Texas Instruments Synchronous Serial Protocol (SSP)

Data transfers begin by asserting the frame indicator line (`ss_0_n/ss_in_n`) for one serial clock period. Data to be transmitted are driven onto the `txd` line one serial clock cycle later; similarly data from the slave are driven onto the `rxd` line. Data are propagated on the rising edge of the serial clock (`sclk_out/sclk_in`) and captured on the falling edge. The length of the data frame ranges from 4 to 16/32-bits (depending upon `SSI_MAX_XFER_SIZE`).

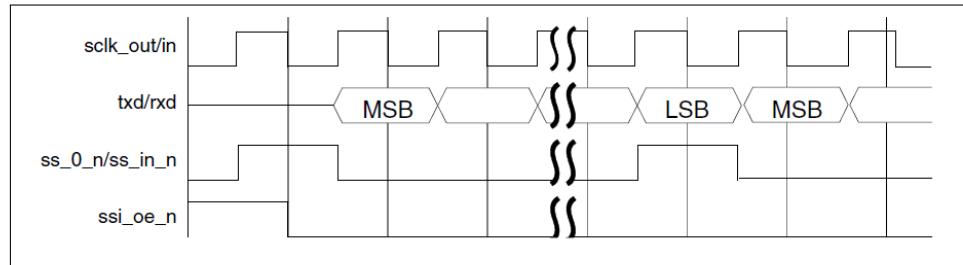
Figure 19.23 shows the timing diagram for a single SSP serial transfer.



**Figure 19.23:** SSP Serial Format

Continuous data frames are transferred in the same way as single data frames. The frame indicator is asserted for one clock period during the same cycle as the LSB from the current transfer, indicating that another data frame follows.

Figure 19.24 shows the timing for a continuous SSP transfer.



**Figure 19.24:** SSP Serial Format Continuous Transfer

### 19.2.6.3 Enhanced SPI Modes

DW\_apb\_ssi supports the dual, quad, and octal modes of SPI using the SSI\_SPI\_MODE configuration parameter. The possible values for this parameter are Standard, Dual SPI, Quad SPI and Octal SPI modes. When dual, quad, or octal mode is selected for this parameter, the width of txd, rxd and ssi\_oe\_n signals change to 2, 4, or 8, respectively. Hence, the data is shifted out/in on more than one line, increasing the overall throughput. All four combinations of the serial clock's polarity and phase are valid in this mode and it works same as in normal SPI mode as described in “Motorola Serial Peripheral Interface (SPI)” on page 742. Dual SPI, Quad or Octal SPI modes function similarly except for the width of txd, rxd and ssi\_oe\_n signals. The mode of operation (write/read) can be selected using the CTRLR0.TMOD field.

#### 19.2.6.3.1 Write Operation in Enhanced SPI Modes

Dual, Quad, or Octal SPI write operations can be divided into three parts:

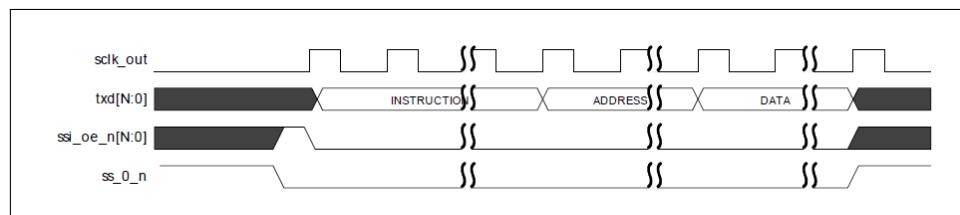
- Instruction phase
- Address phase
- Data phase

The following register fields are used for a write operation:

- CTRLR0.SPI\_FRF - Specifies the format in which the transmission happens for the frame.
- SPI\_CTRLR0 (Control Register 0 register) Specifies length of instruction, address, and data.
- SPI\_CTRLR0.INST\_L Specifies length of an instruction (possible values for an instruction are 0, 4, 8, or 16 bits.)
- SPI\_CTRLR0.ADDR\_L Specifies address length (See Table 19.1 on page 752 for decode values)
- CTRLR0.DFS or CTRLR0.DFS\_32 Specifies data length.

An instruction takes one FIFO location and address can take more than one FIFO locations. Both the instruction and address must be programmed in the data register (DR). DW\_apb\_ssi will wait until both have been programmed to start the write operation. The instruction, address and data can be programmed to send in dual/quad mode, which can be selected from the SPI\_CTRLR0.TRANS\_TYPE and CTRLR0.SPI\_FRF fields.

Figure 19.25 shows a typical write operation in Dual, Quad, or Octal SPI Mode. The value of N will be: 7 if SSI\_SPI\_MODE is set to 3, 3 if SSI\_SPI\_MODE is set to 2, and 1 if SSI\_SPI\_MODE is set to 1. For one write operation, the instruction and address are sent only once followed by data frames programmed in DR until the transmit FIFO becomes empty.

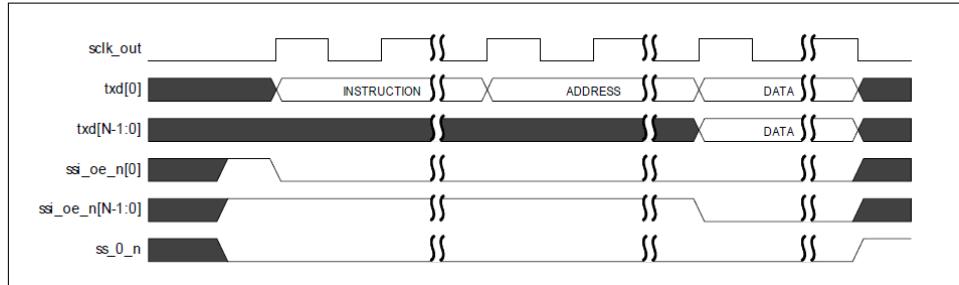


**Figure 19.25:** Typical Write Operation Dual/Quad/Octal SPI Mode

To initiate a Dual/Quad/Octal write operation, CTRLR0.SPI\_FRF must be set to 01/10/11, respectively. This will set the transfer type, and for each write command, data will be transferred in the format specified in CTRLR0.SPI\_FRF field. Following are the possible cases of write operation in enhanced SPI modes

- Case A: Instruction and address both transmitted in standard SPI format

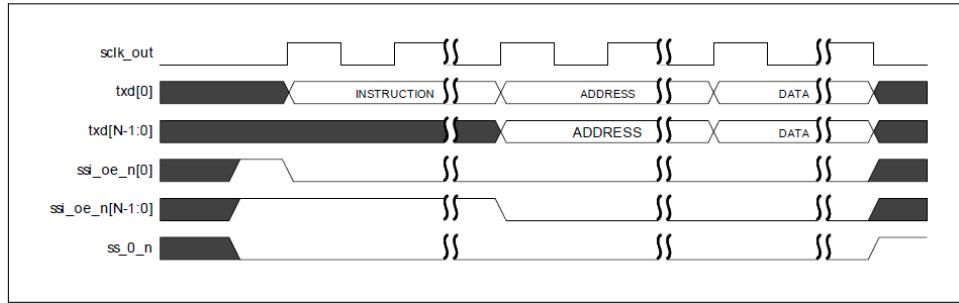
For this, SPI\_CTRLR0.TRANS\_TYPE field must be set to 00. Figure 19.26 show the timing diagram when both instruction and address are transmitted in standard SPI format. The value of N will be: 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.26:** Instruction and Address Transmitted in Standard SPI Format

- Case B: Instruction transmitted in standard and address transmitted in Enhanced SPI format

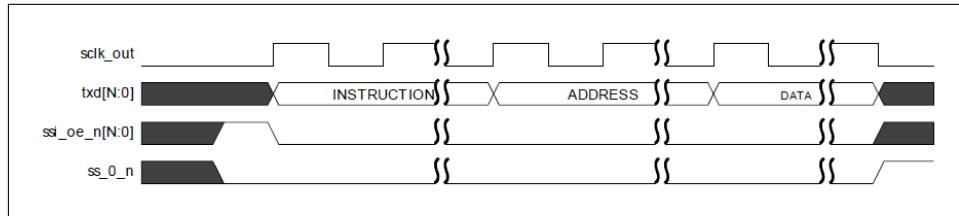
For this, SPI\_CTRLR0.TRANS\_TYPE field must be set to 01. Figure 19.27 shows the timing diagram when an instruction is transmitted in standard format and address is transmitted in dual SPI format specified in the CTRLR0.SPI\_FRF field. The value of N will be: 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.27:** Instruction Transmitted in Standard and Address Transmitted in Enhanced SPI Format

- Case C: Instruction and Address both transmitted in Enhanced SPI format

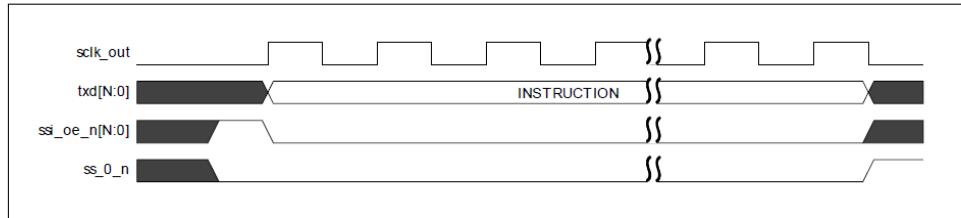
For this, SPI\_CTRLR0.TRANS\_TYPE field must be set to 10. Figure 19.28 shows the timing diagram in which instruction and address are transmitted in SPI format specified in the CTRLR0.SPI\_FRF field. The value of N will be: 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.28:** Instruction and Address Both Transmitted in Enhanced SPI Format

- Case D: Instruction only transfer in enhanced SPI format

For this, SPI\_CTRLR0.TRANS\_TYPE field must be set to 10. Figure 19.29 shows the timing diagram for such a transfer. The value of N will be: 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.29:** Instruction only transfer in enhanced SPI Format

#### 19.2.6.3.2 Read Operation in Enhanced SPI Modes

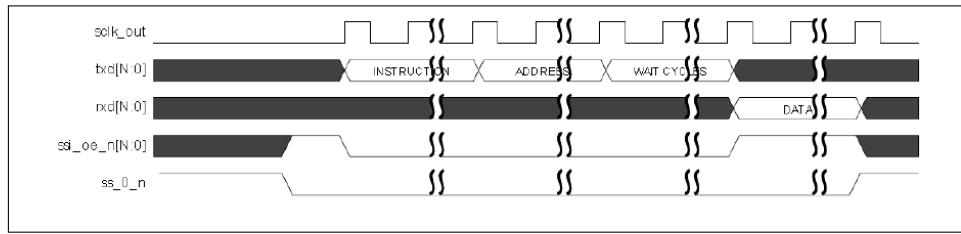
A Dual, Quad, or Octal SPI read operation can be divided into four phases:

- Instruction phase
- Address phase
- Wait cycles
- Data phase

Wait Cycles can be programmed using SPI\_CTRLR0.WAIT\_CYCLES field. The value programmed into SPI\_CTRLR0.WAIT\_CYCLES is mapped directly to sclk\_out times. For example, WAIT\_CYCLES=0 indicates no Wait, WAIT\_CYCLES=1, indicates 1 wait cycle and so on. The wait cycles are introduced for target slave to change their mode from input to output and the wait cycles can vary for different devices.

For a READ operation, DW\_apb\_ssi sends instruction and control data once and waits until it receives NDF (CTRLR1 register) number of data frames and then de-asserts slave select signal.

Figure 19.30 shows a typical read operation in dual quad SPI mode. The value of N will be: 7 if SSI\_SPI\_MODE is set to Octal mode, 3 if SSI\_SPI\_MODE is set to Quad mode, and 1 if SSI\_SPI\_MODE is set to Dual mode.



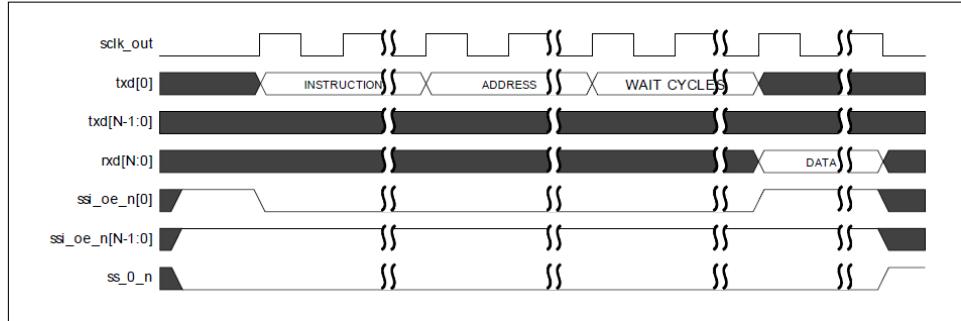
**Figure 19.30:** Typical Read Operation in Enhanced SPI Mode

To initiate a dual/quad/octal read operation, CTRLR0.SPI\_FRF must be set to 01/10/11 respectively. This will set the transfer type, now for each read command data will be transferred in the format specified in CTRLR0.SPI\_FRF field.

Following are the possible cases of write operation in enhanced SPI modes:

- Case A: Instruction and address both transmitted in standard SPI format

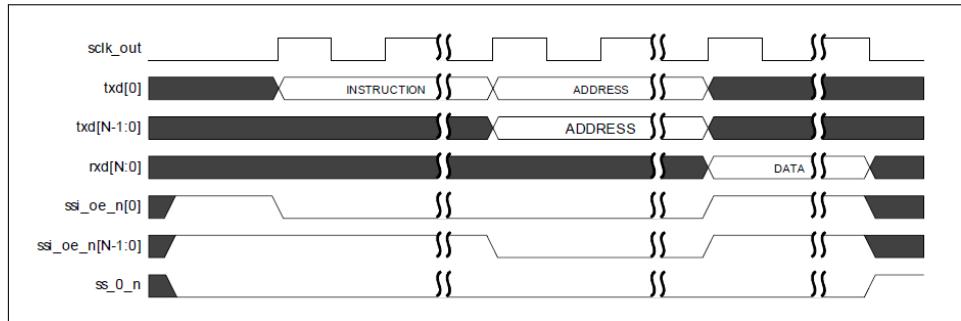
For this, SPI\_CTRLR0.TRANS\_TYPE field should be set to 00. Figure 19.31 shows the timing diagram when both instruction and address are transferred in standard SPI format. The figure also shows WAIT cycles after address, which can be programmed in the SPI\_CTRLR0.WAIT\_CYCLES field. The value of N will be 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.31:** Instruction and Address Transmitted in Standard SPI Format

- Case B: Instruction transmitted in standard and address transmitted in dual SPI format

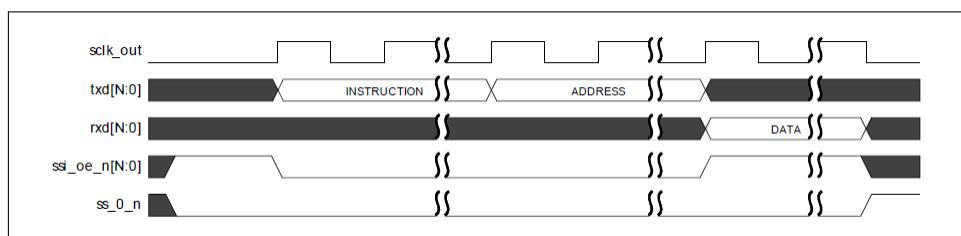
For this, SPI\_CTRLR0.TRANS\_TYPE field should be set to 01. Figure 19.32 shows the timing diagram in which instruction is transmitted in standard format and address is transmitted in dual SPI format. The value of N will be 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.32:** Instruction Transmitted in Standard and Address Transmitted in Enhanced SPI Format

- Case C: Instruction and Address both transmitted in Dual SPI format

For this, SPI\_CTRLR0.TRANS\_TYPE field must be set to 10. Figure 19.33 shows the timing diagram in which both instruction and address are transmitted in dual SPI format. The value of N will be: 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.33:** Instruction and Address Transmitted in Enhanced SPI Format

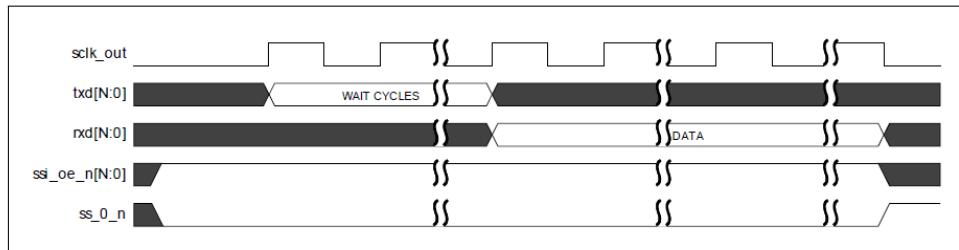
- Case D: No Instruction, No Address READ transfer

For this, SPI\_CTRLR0.ADDR\_L and SPI\_CTRLR0.INST\_L must be set to 0 and SPI\_CTRLR0.WAIT\_CYCLES must be set to a non-zero value. Table 19.1 lists the ADDR\_L decode value and the respective description for enhanced (Dual/Quad/Octal) SPI modes.

**Table 19.1:** ADDR\_L Decode in Enhanced SPI Mode

ADDR_L Decode Value	Description
0000	0-bit Address Width
0001	4-bit Address Width
0010	8-bit Address Width
0011	12-bit Address Width
0100	16-bit Address Width
0101	20-bit Address Width
0110	24-bit Address Width
0111	28-bit Address Width
1000	32-bit Address Width
1001	36-bit Address Width
1010	40-bit Address Width
1011	44-bit Address Width
1100	48-bit Address Width
1101	52-bit Address Width
1110	56-bit Address Width
1111	60-bit Address Width

Figure 19.34 shows the timing diagram for such type of transfer. The value of N will be: 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01. To initiate this transfer, the software has to perform a dummy write in the data register (DR), DW\_apb\_ssi will wait for programmed wait cycles and then fetch the amount of data specified in NDF field.

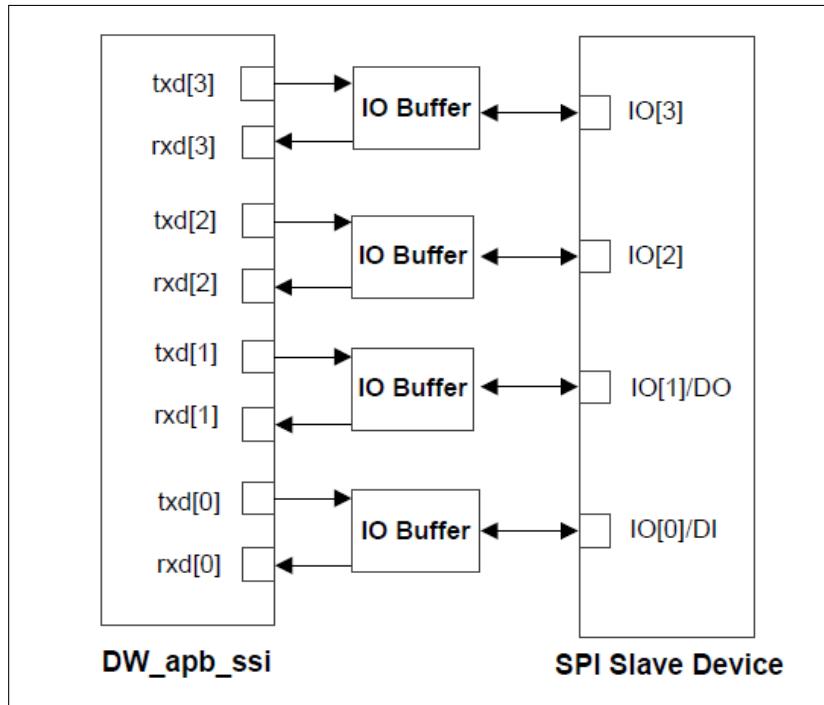


**Figure 19.34:** No Instruction and No Address READ Transfer

**19.2.6.3.3 Advanced I/O Mapping for Enhanced SPI Modes** The Input/Output mapping for enhanced SPI modes (dual, quad, and octal) is configurable using the SSI\_IO\_EN parameter, which configures whether the I/O mapping of a slave device is hardcoded inside the DW\_apb\_ssi. When SSI\_IO\_MAP\_EN is set to 1, the rxd[1] signal will be used to sample incoming data in standard SPI mode of operation.

For other protocols (such as SSP and Microwire), the I/O mapping remains the same. Therefore, it is easy for other protocols to connect with any device that supports Dual/Quad SPI operation because other protocols do not require a MUX logic to exist outside the design.

Figure 19.35 shows the I/O mapping of DW\_apb\_ssi in Quad mode with another SPI device that supports the Quad mode. As illustrated in Figure 19.35, the IO[1] pin is used as DO in standard SPI mode of operation and it is connected to rxd[1] pin, which will be sampling the input in the standard mode of operation.



**Figure 19.35:** Advanced I/O Mapping in Quad SPI Modes

#### 19.2.6.4 Dual Data-Rate (DDR) Support in SPI Operation

In standard operations, data transfer in SPI modes occur on either the positive or negative edge of the clock. For improved throughput, the dual data-rate transfer can be used for reading or writing to the memories.

The DDR mode supports the following modes of SPI protocol:

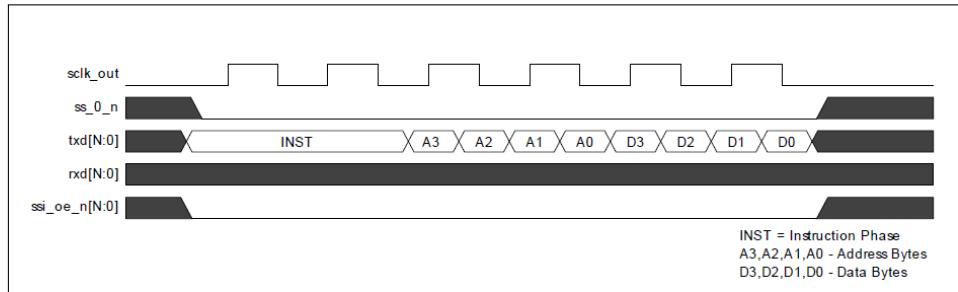
- SCPH=0 & SCPOL=0 (Mode 0)
- SCPH=1 & SCPOL=1 (Mode 3)

DDR commands enable data to be transferred on both edges of clock. Following are the different types of DDR commands:

- Address and data are transmitted (or received in case of data) in DDR format, while instruction is transmitted in standard format.
- Instruction, address, and data are all transmitted or received in DDR format.

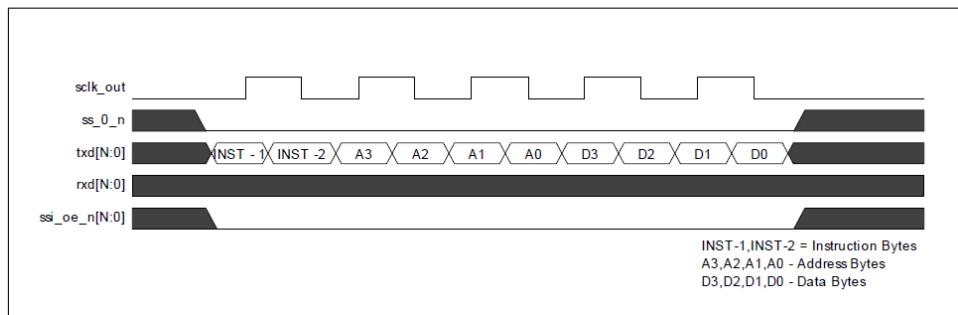
The DDR\_EN (SPI\_CTRLR0[16]) bit is used to determine if the Address and data have to be transferred in DDR mode and INST\_DDR\_EN (SPI\_CTRLR0[17]) bit is used to determine if Instruction must be transferred in DDR format. These bits are only valid when the CTRLR0.SPI\_FRF bit is set to be in Dual, Quad or Octal mode.

Figure 19.36 describes a DDR write transfer where instructions are continued to be transmitted in standard format. In Figure 19.36, the value of N will be 7 if CTRLR0.SPI\_FRF is set to 11, 3 if CTRLR0.SPI\_FRF is set to 10, and 1 if CTRLR0.SPI\_FRF is set to 01.



**Figure 19.36:** DDR Transfer with SCPH=0 and SCPOL=0

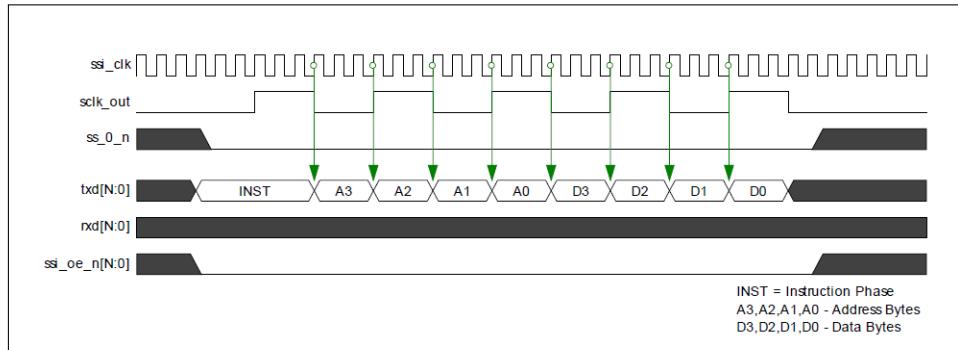
Figure 19.37 describes a DDR write transfer where instruction, address and data all are transferred in DDR format.



**Figure 19.37:** DDR Transfer with Instruction, Address and Data Transferred in DDR Format

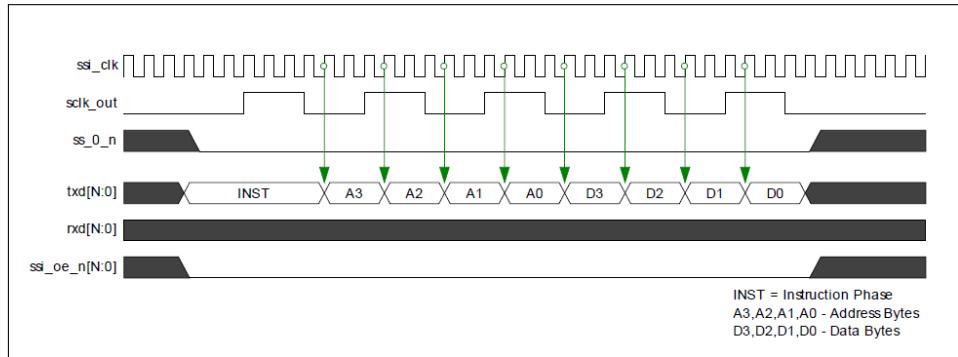
**19.2.6.4.1 Transmitting Data in DDR Mode** In DDR mode, data is transmitted on both edges so that it is difficult to sample data correctly. DW\_apb\_ssi uses an internal register to determine the edge on which the data should be transmitted. This will ensure that the receiver is able to get a stable data while sampling. The internal register (DDR\_DRIVE\_EDGE) determines the edge on which the data is transmitted. DW\_apb\_ssi sends data with respect to baud clock, which is an integral multiple of the internal clock ( $ssi\_clk * BAUDR$ ). The data needs to be transmitted within half clock cycle ( $BAUDR/2$ ), therefore the maximum value for DDR\_DRIVE\_EDGE is equal to  $[(BAUDR/2)-1]$ . If the programmed value of DDR\_DRIVE\_EDGE is 0 then data is transmitted edge-aligned with respect to sclk\_out (baud clock). If the programmed value of DDR\_DRIVE\_EDGE is 1 then the data is transmitted one  $ssi\_clk$  before the edge of sclk\_out.

Figure 19.38, Figure 19.39, and Figure 19.40 show examples of how data is transmitted using different values of the DDR\_DRIVE\_EDGE register. The green arrows in these examples represent the points where data is driven. Baud rate used in all these examples is 12. In Figure 19.38, transmit edge and driving edge of the data are the same. This is default behavior in DDR mode.

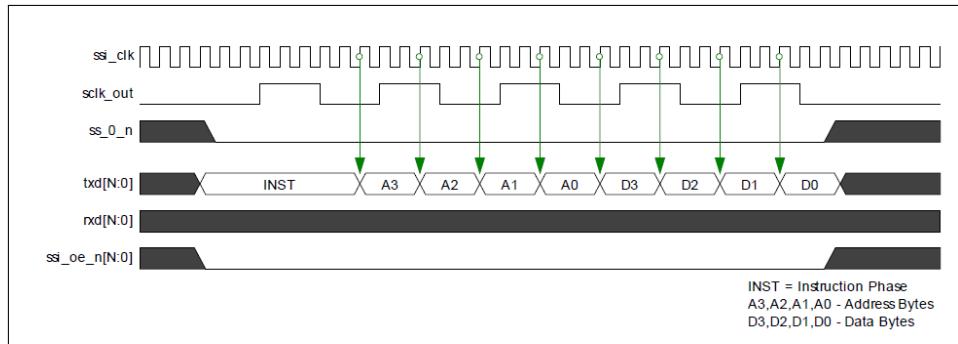


**Figure 19.38:** Transmit Data With `DDR_DRIVE_EDGE = 0`

Figure 19.38 shows the default behavior in which the transmit and driving edge of the data is the same.



**Figure 19.39:** Transmit Data With `DDR_DRIVE_EDGE = 1`



**Figure 19.40:** Transmit Data With DDR\_DRIVE\_EDGE = 2

#### 19.2.6.5 XIP Mode Support in SPI Mode

The eXecute In Place (XIP) mode enables transfer of SPI data directly through the APB interface without writing the data register of DW\_apb\_ssi. XIP mode can be enabled in DW\_apb\_ssi by selecting the SSI\_XIP\_EN configuration parameter, which includes an extra sideband signal `xip_en`, on the APB interface. This signal indicates whether APB transfers are register read-write or XIP reads. If the `xip_en` signal is driven to 1, DW\_apb\_ssi expects only read request on the APB interface. This request is translated to SPI read on the serial interface and soon after the data is received, the data is returned to the APB interface in the same transaction.

The `paddr` signal is used to derive the address to be sent on the XIP interface. The address length is derived from the SPI\_CTRLR0.ADDR\_L field, and relevant bits from `paddr` ([SPI\_CTRLR0.ADDR\_L-1:0]) are transferred as address to the SPI interface.

**19.2.6.5.1 Read Operation in XIP Mode** The XIP operation is supported only in enhanced SPI modes (Dual, Quad, and Octal) of operation. Therefore, the CTRLR0.SPI\_FRF bit should not be programmed to 0. An XIP read operation is divided into two phases:

- Address phase
- Data phase

For an XIP read operation

1. Set the SPI frame format and data frame size value in CTRLR0 register. Note that the value of the maximum data frame size is equal to the APB\_DATA\_WIDTH parameter.
2. Set the Address length, Wait cycles, and transaction type in the SPI\_CTRLR0 register. Note that the maximum address length is 32.

After these settings, a user can initiate a read transaction through the APB interface which will be transferred to SPI peripheral using programmed values. Figure 19.41 shows the typical XIP transfer. The Value of N = 1, 3 and 7 for SPI mode Dual, Quad and Octal modes, respectively.

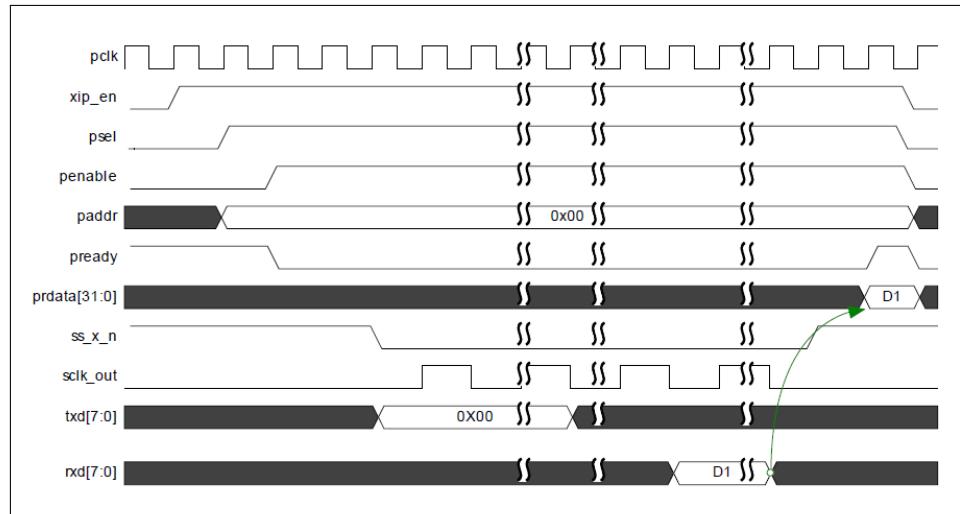


Figure 19.41: Typical Read Operation in XIP Mode

### 19.2.7 DMA Controller Interface

The DW\_apb\_ssi has optional built-in DMA capability which can be selected at configuration time; it has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA. While the DW\_apb\_ssi DMA operation is designed in a generic way to fit any DMA controller as easily as possible, it is designed to work seamlessly, and best used, with the DesignWare DMA Controller, the DW\_ahb\_dmac. The settings of the DW\_ahb\_dmac that are relevant to the operation of the DW\_apb\_ssi are discussed here, mainly bit fields in the DW\_ahb\_dmac channel control register, CTLx, where x is the channel number.

The DW\_apb\_ssi uses two DMA channels, one for the transmit data and one for the receive data. The DW\_apb\_ssi has these DMA registers:

- DMACR Control register to enable DMA operation.
- DMATDLR Register to set the transmit the FIFO level at which a DMA request is made.
- DMARDLR Register to set the receive FIFO level at which a DMA request is made.

The DW\_apb\_ssi uses the following handshaking signals to interface with the DMA controller.

- dma\_tx\_req
- dma\_tx\_single
- dma\_tx\_ack
- dma\_rx\_req
- dma\_rx\_single
- dma\_rx\_ack

They are discussed further in the “Handshaking Interface Operation” on page 764.

The DMA output dma\_finish is a status signal to indicate that the DMA block transfer is complete; for more information on the dma\_finish signal, refer to the Signals chapter in the DesignWare DW\_ahb\_dmac Databook. The DW\_apb\_ssi does not use this status signal, and therefore it does not appear in the I/O signal list.

To enable the DMA Controller interface on the DW\_apb\_ssi, you must write the DMA Control Register (DMACR). Writing a 1 into the TDMAE bit field of DMACR register enables the DW\_apb\_ssi transmit handshaking interface. Writing a 1 into the RDMAE bit field of the DMACR register enables the DW\_apb\_ssi receive handshaking interface.

Table 19.2 provides description for different DMA transmit data level values.

DMATDL Value	Description
0000_0000	dma_tx_req is asserted when 0 data entries are present in the transmit FIFO
0000_0001	dma_tx_req is asserted when 1 or less data entry is present in the transmit FIFO
0000_0010	dma_tx_req is asserted when 2 or less data entries are present in the transmit FIFO
0000_0011	dma_tx_req is asserted when 3 or less data entries are present in the transmit FIFO
...	...
...	...
1111_1100	dma_tx_req is asserted when 252 data entries are present in the transmit FIFO
1111_1101	dma_tx_req is asserted when 253 data entries are present in the transmit FIFO
1111_1110	dma_tx_req is asserted when 254 data entries are present in the transmit FIFO
1111_1111	dma_tx_req is asserted when 255 data entries are present in the transmit FIFO

**Table 19.2:** DMA Transmit Data Level (DMATDL) Decode Value

Table 19.3 provides description for different DMA Receive Data Level values.

DMARDL Value	Description
0000_0000	dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO
0000_0001	dma_rx_req is asserted when 2 or more data entries are present in the receive FIFO
0000_0010	dma_rx_req is asserted when 3 or more data entries are present in the receive FIFO
0000_0011	dma_rx_req is asserted when 4 or more data entries are present in the receive FIFO
...	...
...	...
1111_1100	dma_rx_req is asserted when 253 more data entries are present in the receive FIFO
1111_1101	dma_rx_req is asserted when 254 more data entries are present in the receive FIFO
1111_1110	dma_rx_req is asserted when 255 more data entries are present in the receive FIFO
1111_1111	dma_rx_req is asserted when 256 data entries are present in the receive FIFO

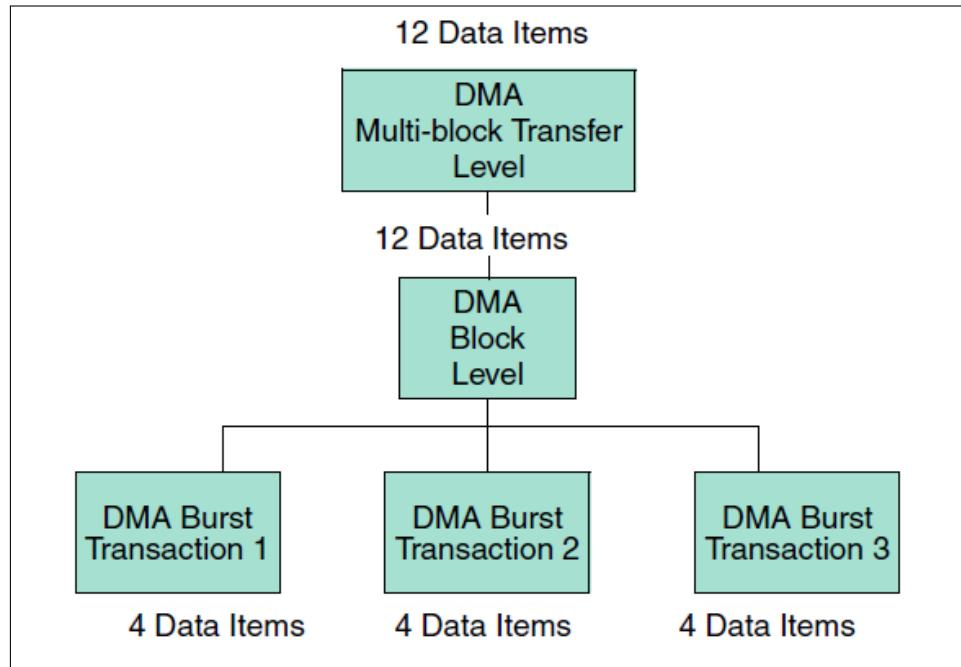
**Table 19.3:** DMA Receive Data Level (DMARDL) Decode Value

### 19.2.7.1 Overview of Operation

As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by the DW\_apb\_ssi; this is programmed into the BLOCK\_TS field of the CTLx register.

The block is broken into a number of transactions, each initiated by a request from the DW\_apb\_ssi. The DMA Controller must also be programmed with the number of data items (in this case, DW\_apb\_ssi FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length, and is programmed into the SRC\_MSIZE/DEST\_MSIZE fields of the DW\_ahb\_dmac CTLx register for source and destination, respectively.

Figure 19.42 shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4. In this case, the block size is a multiple of the burst transaction length; therefore, the DMA block transfer consists of a series of burst transactions.



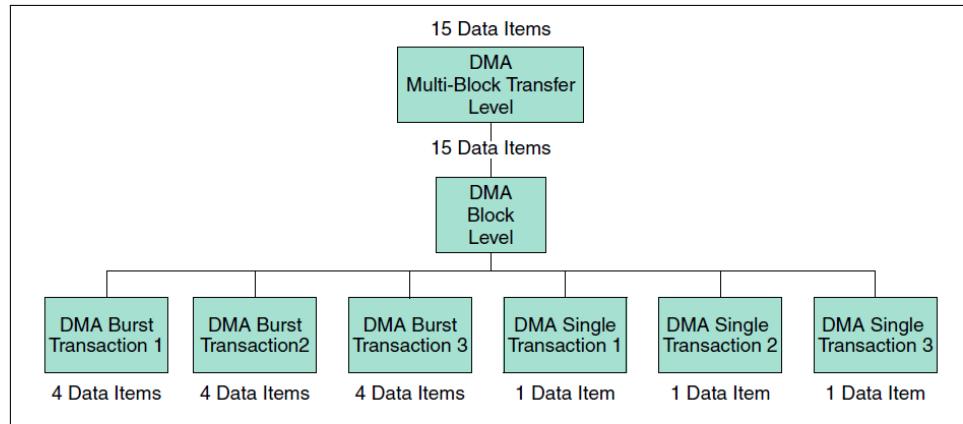
**Figure 19.42:** Burst Transaction  $pclk = hclk$

If the DW\_apb\_ssi makes a transmit request to this channel, four data items are written to the DW\_apb\_ssi transmit FIFO. Similarly, if the DW\_apb\_ssi makes a receive request to this channel, four data items are read from the DW\_apb\_ssi receive FIFO. Three separate requests must be made to this DMA channel before all 12 data items are written or read.

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in Figure 19.43, a series of burst transactions followed by single transactions are needed to complete the block transfer.

### 19.2.7.2 Transmit Watermark Level and Transmit FIFO Underflow

During DW\_apb\_ssi serial transfers, transmit FIFO requests are made to the DW\_ahb\_dmac whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (DMATDLR) value;



**Figure 19.43:** Burst Transaction  $\text{pclk} = \text{hclk}$

this is known as the watermark level. The DW\_ahb\_dmac responds by writing a burst of data to the transmit FIFO buffer, of length  $\text{CTLx.DEST\_MSIZE}$ .

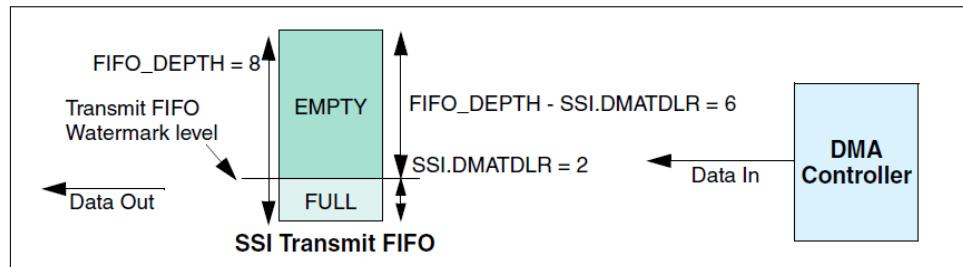
Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise the FIFO will run out of data (underflow). To prevent this condition, the user must set the watermark level correctly.

#### 19.2.7.3 Choosing the Transmit Watermark Level

Consider the example where the assumption is made:

$$\text{DMA.CTLx.DEST\_MSIZE} = \text{FIFO\_DEPTH} - \text{SSI.DMATDLR}$$

Here the number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.



**Figure 19.44:** Burst Transaction  $\text{pclk} = \text{hclk}$

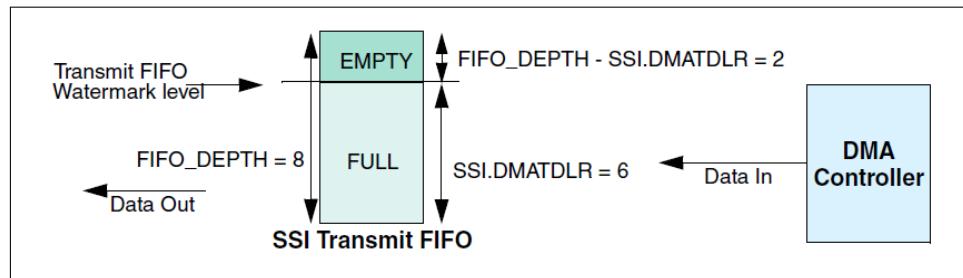
##### 19.2.7.3.1 Case 1: $\text{DMATDLR} = 2$

- Transmit FIFO watermark level =  $\text{SSI.DMATDLR} = 2$
- $\text{DMA.CTLx.DEST\_MSIZE} = \text{FIFO\_DEPTH} - \text{SSI.DMATDLR} = 6$
- $\text{SSI transmit FIFO\_DEPTH} = 8$
- $\text{DMA.CTLx.BLOCK\_TS} = 30$

Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

$$\text{DMA.CTLx.BLOCK_TS}/\text{DMA.CTLx.DEST_MSIZ} = 30/6 = 5$$

The number of burst transactions in the DMA block transfer is 5. But the watermark level, SSI.DMATDLR, is quite low. Therefore, the probability of an SSI underflow is high where the SSI serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.



**Figure 19.45:** Burst Transaction pclk = hclk

#### 19.2.7.3.2 Case 2: DMATDLR = 6

- Transmit FIFO watermark level = SSI.DMATDLR = 6
- DMA.CTLx.DEST\_MSIZ = FIFO\_DEPTH - SSI.DMATDLR = 2
- SSI transmit FIFO\_DEPTH = 8
- DMA.CTLx.BLOCK\_TS = 30

Number of burst transactions in Block:

$$\text{DMA.CTLx.BLOCK_TS}/\text{DMA.CTLx.DEST_MSIZ} = 30/2 = 15$$

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level, SSI.DMATDLR, is high. Therefore, the probability of an SSI underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the SSI transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of AMBA bursts per block and worse bus utilization than the former case.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the SSI transmits data to the rate at which the DMA can respond to destination burst requests.

For example, promoting the channel to the highest priority channel in the DMA, and promoting the DMA master interface to the highest priority master in the AMBA layer, increases the rate at which the DMA controller can respond to burst transaction requests. This in turn allows the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

#### 19.2.7.4 Selecting DEST\_MSIZEx and Transmit FIFO Overflow

As can be seen from Figure 19.45 on page 762, programming DMA.CTLx.DEST\_MSIZEx to a value greater than the watermark level that triggers the DMA request may cause overflow when there is not enough space in the SSI transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow:

$$\text{DMA.CTLx.DEST_MSIZEx} \leq \text{SSI.FIFO_DEPTH} - \text{SSI.DMATDLR} \quad (1)$$

In Case 2: DMATDLR = 6, the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST\_MSIZEx. Thus, the transmit FIFO may be full, but not overflowed, at the completion of the burst transaction.

Therefore, for optimal operation, DMA.CTLx.DEST\_MSIZEx should be set at the FIFO level that triggers a transmit DMA request; that is:

$$\text{DMA.CTLx.DEST_MSIZEx} = \text{SSI.FIFO_DEPTH} - \text{SSI.DMATDLR} \quad (2)$$

This is the setting used in Figure 19.43 on page 761.

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, and this in turn improves AMBA bus utilization.

#### 19.2.7.5 Receive Watermark Level and Receive FIFO Overflow

During DW\_apb\_ssi serial transfers, receive FIFO requests are made to the DW\_ahb\_dmac whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register; that is, DMARDLR+1. This is known as the watermark level. The DW\_ahb\_dmac responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC\_MSIZEx.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO will fill with data (overflow). To prevent this condition, the user must correctly set the watermark level.

#### 19.2.7.6 Choosing the Receive Watermark level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, DMARDLR+1, should be set to minimize the probability of overflow, as shown in Figure 19.46. It is a tradeoff between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

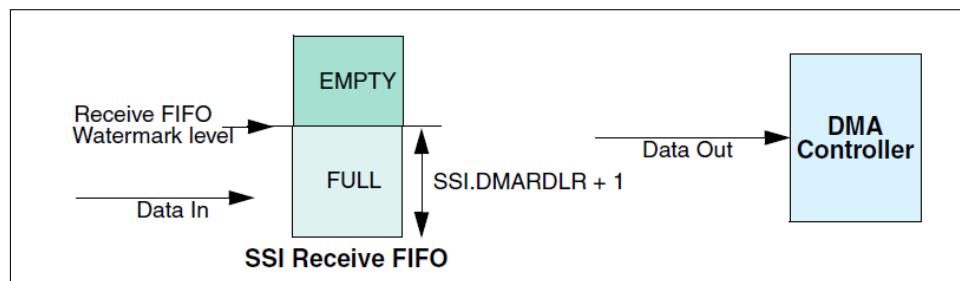


Figure 19.46: Burst Transaction pclk = hclk

### 19.2.7.7 Selecting SRC\_MSIZEx and Receive FIFO Underflow

As seen in Figure 19.46, programming a source burst transaction length greater than the watermark level may cause underflow when there is not enough data to service the source burst request. Therefore, equation (3) below must be adhered to avoid underflow.

If the number of data items in the receive FIFO is equal to the source burst length at the time the burst request is made DMA.CTLx.SRC\_MSIZEx the receive FIFO may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC\_MSIZEx should be set at the watermark level; that is: DMA.CTLx.SRC\_MSIZEx = SSI.DMARDLR + 1 (3)

Adhering to equation (3) reduces the number of DMA bursts in a block transfer, and this in turn can improve AMBA bus utilization.

### 19.2.7.8 Handshaking Interface Operation

The following sections discuss the DW\_apb\_ssi handshaking interface.

**19.2.7.8.1 dma\_tx\_req, dma\_rx\_req** The request signals for source and destination, dma\_tx\_req and dma\_rx\_req, are activated when their corresponding FIFOs reach the watermark levels as discussed earlier.

The DW\_ahb\_dmac uses rising-edge detection of the dma\_tx\_req/dma\_rx\_req to identify a request on the channel. Upon reception of the dma\_tx\_ack/dma\_rx\_ack signal from the DW\_ahb\_dmac to indicate the burst transaction is complete, the DW\_apb\_ssi de-asserts the burst request signals, dma\_tx\_req/dma\_rx\_req, until dma\_tx\_ack/dma\_rx\_ack is de-asserted by the DW\_ahb\_dmac.

When the DW\_apb\_ssi samples that dma\_tx\_ack/dma\_rx\_ack is de-asserted, it can re-assert the dma\_tx\_req/dma\_rx\_req of the request line if their corresponding FIFOs exceed their watermark levels (back-to-back burst transaction). If this is not the case, the DMA request lines remain de-asserted.

Figure 19.47 on page 764 shows a timing diagram of a burst transaction where pclk = hclk.

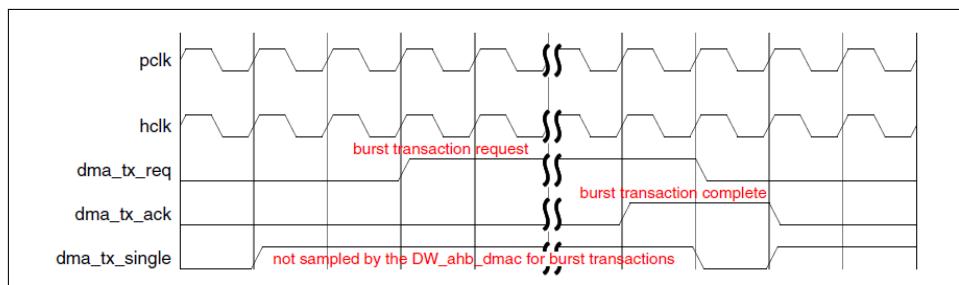
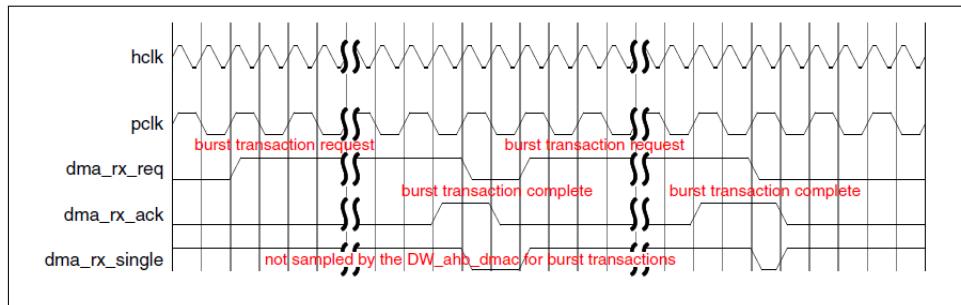


Figure 19.47: Burst Transaction pclk = hclk

Figure 19.48 shows two back-to-back burst transactions where the hclk frequency is twice the pclk frequency.



**Figure 19.48:** Back-to-Back Burst Transactions  $hclk = 2 * pclk$

The handshaking loop is as follows :

dma\_tx\_req/dma\_rx\_req asserted by DW\_apb\_ssi

- dma\_tx\_ack/dma\_rx\_ack asserted by DW\_ahb\_dmac
- dma\_tx\_req/dma\_rx\_req de-asserted by DW\_apb\_ssi
- dma\_tx\_ack/dma\_rx\_ack de-asserted by DW\_ahb\_dmac
- dma\_tx\_req/dma\_rx\_req re-asserted by DW\_apb\_ssi, if back-to-back transaction is required

Two things to keep in mind:

- The burst request lines, dma\_tx\_req signal/dma\_rx\_req, once asserted remain asserted until their corresponding dma\_tx\_ack/dma\_rx\_ack signal is received even if the respective FIFOs drop below their watermark levels during the burst transaction.
- The dma\_tx\_req/dma\_rx\_req signals are de-asserted when their corresponding dma\_tx\_ack/dma\_rx\_ack signals are asserted, even if the respective FIFOs exceed their watermark levels.

**19.2.7.8.2 dma\_tx\_single, dma\_rx\_single** The dma\_tx\_single signal is asserted when there is at least one free entry in the transmit FIFO, and is cleared when the dma\_tx\_ack signal is active. The dma\_tx\_single signal will be re-asserted when the dma\_tx\_ack signal is de-asserted, if the condition for setting still holds true.

The dma\_rx\_single signal is asserted when there is at least one valid data entry in the receive FIFO, and is cleared when the dma\_rx\_ack signal is active. The dma\_rx\_single signal will be re-asserted when the dma\_rx\_ack signal is de-asserted, if the condition for setting still holds true.

These signals are needed by only the DW\_ahb\_dmac for the case where the block size, CTLx.BLOCK\_TS, that is programmed into the DW\_ahb\_dmac is not a multiple of the burst transaction length, CTLx.SRC\_MSIZE, CTLx.DEST\_MSIZE, as shown in Figure 19.43 on page 761. In this case, the DMA single outputs inform the DW\_ahb\_dmac that it is still possible to perform single data item transfers, so it can access all data items in the transmit/receive FIFO and complete the DMA block transfer. The DMA single outputs from the DW\_apb\_ssi are not sampled by the DW\_ahb\_dmac otherwise. This is illustrated in the following example.

Consider first an example where the receive FIFO channel of the DW\_apb\_ssi is as follows:

DMA.CTLx.SRC\_MSIZE = SSI.DMARDL + 1 = 4

DMA.CTLx.BLOCK\_TS = 12

For the example in Figure 19.42 on page 760, with the block size set to 12, the `dma_rx_req` signal is asserted when four data items are present in the receive FIFO. The `dma_rx_req` signal is asserted three times during the DW\_apb\_ssi serial transfer, ensuring that all 12 data items are read by the DW\_ahb\_dmac. All DMA requests read a block of data items and no single DMA transactions are required. This block transfer is made up of three burst transactions.

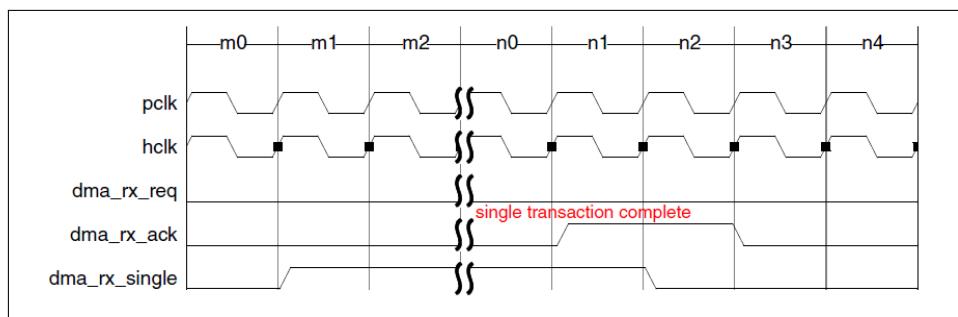
Now, for the following block transfer:

`DMA.CTLx.SRC_MSIZE = SSI.DMARDLR + 1 = 4`

`DMA.CTLx.BLOCK_TS = 15`

The first 12 data items are transferred as already described using 3 burst transactions. But when the last three data frames enter the receive FIFO, the `dma_rx_req` signal is not activated because the FIFO level is below the watermark level. The DW\_ahb\_dmac samples `dma_rx_single` and completes the DMA block transfer using three single transactions. The block transfer is made up of three burst transactions followed by three single transactions.

Figure 19.49 shows a single transaction.



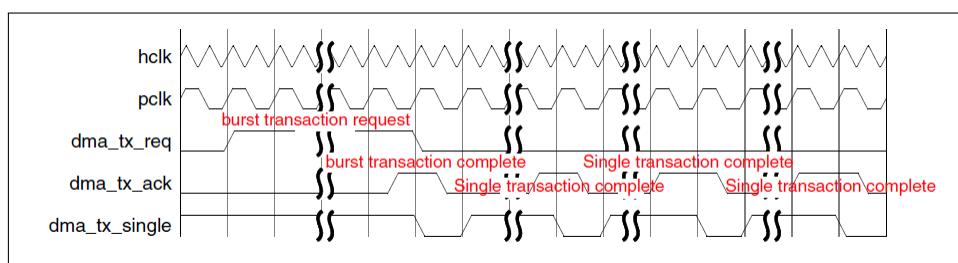
**Figure 19.49:** Single Transaction

The handshaking loop is as follows:

`dma_tx_single/dma_rx_single` asserted by DW\_apb\_ssi

- `dma_tx_ack/dma_rx_ack` asserted by DW\_ahb\_dmac
- `dma_tx_single/dma_rx_single` de-asserted by DW\_apb\_ssi
- `dma_tx_ack/dma_rx_ack` de-asserted by DW\_ahb\_dmac.

Figure 19.50 shows a burst transaction, followed by three back-to-back single transactions, where the `hclk` frequency is twice the `pclk` frequency.



**Figure 19.50:** Burst Transaction + 3 Back-to-Back Singles  $hclk = 2 * pclk$

## 19.3 Register Description

### 19.3.1 Register Map Summary

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)

Register	Offset	Description	Reset Value
CTRLR0	0x00	Control Register0	0x0000_0000
CTRLR1	0x04	Control Register1	0x0000_0000
SSIENR	0x08	SSI Enable Register	0x0000_0000
MWCR	0x0C	Microwire Control Register	0x0000_0000
SER	0x10	Slave Enable Register	0x0000_0000
BAUDR	0x14	Baud Rate Select Register	0x0000_0000
TXFTLR	0x18	Transmit FIFO Threshold Level Register	0x0000_0000
RXFTLR	0x1C	Receive FIFO Threshold Level Register	0x0000_0000
TXFLR	0x20	Transmit FIFO Level Register	0x0000_0000
RXFLR	0x24	Receive FIFO Level Register	0x0000_0000
SR	0x28	Status Register	0x0000_0000
IMR	0x2C	Interrupt Mask Register	0x0000_0000
ISR	0x30	Interrupt Status Register	0x0000_0000
RISR	0x34	Raw Interrupt Status Register	0x0000_0000
TXOICR	0x38	Transmit FIFO Overflow Interrupt Clear Register	0x0000_0000
RXOICR	0x3C	Receive FIFO Overflow Interrupt Clear Register	0x0000_0000
RXUICR	0x40	Receive FIFO Underflow Interrupt Clear Register	0x0000_0000
MSTICR	0x44	Multi-Master Interrupt Clear Register	0x0000_0000
ICR	0x48	Interrupt Clear Register	0x0000_0000
DMACR	0x4C	DMA Control Register	0x0000_0000
DMATDLR	0x50	DMA Transmit Data Level Register	0x0000_0000
DMARDLR	0x54	DMA Receive Data Level Register	0x0000_0000
IDR	0x58	Identification Register	Undefined
SSI_VERSION_ID	0x5C	coreKit version ID Register	Undefined
DRx	0x60	Data Register x(for x =0; x <=35)	0x0000_0000
RX_SAMPLE_DLY	0xF0	RX Sample Delay Register	0x0000_0000
SPI_CTRLR0	0xF4	SPI Control Register	0x0000_0000
TXD_DRIVE_EDGE	0xF8	Transmit Drive Edge Register	0x0000_0000
RSVD	0xFC	Reserved Register	0x0000_0000

### 19.3.2 CTRLR0

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:25]	-	Reserved	-
SSTE	[24]	R/W	<p>Slave Select Toggle Enable.</p> <p>When operating in SPI mode with clock phase (SCPH) set to 0, this register controls the behavior of the slave select line (ss_0_n) between data frames. If this register field is set to 1 the ss_0_n line will toggle between consecutive data frames, with the serial clock (sclk) being held to its default value while ss_0_n is high; if this register field is set to 0 the ss_0_n will stay low and sclk will run continuously for the duration of the transfer.</p>	0
RSVD	[23]	-	Reserved	-
SPI_FRF	[22:21]	R/W	<p>SPI Frame Format:</p> <p>Selects data frame format for Transmitting/Receiving the data</p> <p>Bits only valid when SSI_SPI_MODE is either set to "Dual" or "Quad" or "Octal" mode.</p> <p>When SSI_SPI_MODE is configured for "Dual Mode", 10/11 combination is reserved.</p> <p>When SSI_SPI_MODE is configured for "Quad Mode", 11 combination is reserved.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (STD_SPI_FRF): Standard SPI Frame Format</li> <li>- 0x1 (DUAL_SPI_FRF): Dual SPI Frame Format</li> <li>- 0x2 (QUAD_SPI_FRF): Quad SPI Frame Format</li> <li>- 0x3 (OCTAL_SPI_FRF): Octal SPI Frame Format</li> </ul> <p>Note:</p> <p>SPI0 : Support, SPI1/SPI2 : Not support</p>	0

DFS_32	[20:16]	R/W	<p>Data Frame Size in 32-bit transfer size mode. Used to select the data frame size in 32-bit transfer mode. These bits are only valid when SSI_MAX_XFER_SIZE is configured to 32. When the data frame size is programmed to be less than 32 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You are responsible for making sure that transmit data is right-justified before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data.</p> <p>Note: When SSI_SPI_MODE is either set to "Dual" or "Quad" or "Octal" mode and SPI_FRF is not set to 2'b00.</p> <ul style="list-style-type: none"> <li>- DFS value should be multiple of 2 if SPI_FRF = 0x01,</li> <li>- DFS value should be multiple of 4 if SPI_FRF = 0x10,</li> <li>- DFS value should be multiple of 8 if SPI_FRF = 0x11.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x3 (FRAME_04BITS): 4-bit serial data transfer</li> <li>- 0x4 (FRAME_05BITS): 5-bit serial data transfer</li> <li>- 0x5 (FRAME_06BITS): 6-bit serial data transfer</li> <li>- 0x6 (FRAME_07BITS): 7-bit serial data transfer</li> <li>- 0x7 (FRAME_08BITS): 8-bit serial data transfer</li> <li>- 0x8 (FRAME_09BITS): 9-bit serial data transfer</li> <li>- 0x9 (FRAME_10BITS): 10-bit serial data transfer</li> <li>- 0xa (FRAME_11BITS): 11-bit serial data transfer</li> <li>- 0xb (FRAME_12BITS): 12-bit serial data transfer</li> <li>- 0xc (FRAME_13BITS): 13-bit serial data transfer</li> <li>- 0xd (FRAME_14BITS): 14-bit serial data transfer</li> <li>- 0xe (FRAME_15BITS): 15-bit serial data transfer</li> <li>- 0xf (FRAME_16BITS): 16-bit serial data transfer</li> <li>- 0x10 (FRAME_17BITS): 17-bit serial data transfer</li> <li>- 0x11 (FRAME_18BITS): 18-bit serial data transfer</li> <li>- 0x12 (FRAME_19BITS): 19-bit serial data transfer</li> <li>- 0x13 (FRAME_20BITS): 20-bit serial data transfer</li> <li>- 0x14 (FRAME_21BITS): 21-bit serial data transfer</li> <li>- 0x15 (FRAME_22BITS): 22-bit serial data transfer</li> <li>- 0x16 (FRAME_23BITS): 23-bit serial data transfer</li> <li>- 0x17 (FRAME_24BITS): 24-bit serial data transfer</li> <li>- 0x18 (FRAME_25BITS): 25-bit serial data transfer</li> <li>- 0x19 (FRAME_26BITS): 26-bit serial data transfer</li> <li>- 0x1a (FRAME_27BITS): 27-bit serial data transfer</li> <li>- 0x1b (FRAME_28BITS): 28-bit serial data transfer</li> <li>- 0x1c (FRAME_29BITS): 29-bit serial data transfer</li> <li>- 0x1d (FRAME_30BITS): 30-bit serial data transfer</li> <li>- 0x1e (FRAME_31BITS): 31-bit serial data transfer</li> <li>- 0x1f (FRAME_32BITS): 32-bit serial data transfer</li> </ul>	0
CFS	[15:12]	R/W	<p>Control Frame Size. Selects the length of the control word for the Microwire frame format.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (SIZE_01_BIT): 1-bit Control Word</li> <li>- 0x1 (SIZE_02_BIT): 2-bit Control Word</li> <li>- 0x2 (SIZE_03_BIT): 3-bit Control Word</li> <li>- 0x3 (SIZE_04_BIT): 4-bit Control Word</li> <li>- 0x4 (SIZE_05_BIT): 5-bit Control Word</li> <li>- 0x5 (SIZE_06_BIT): 6-bit Control Word</li> <li>- 0x6 (SIZE_07_BIT): 7-bit Control Word</li> <li>- 0x7 (SIZE_08_BIT): 8-bit Control Word</li> <li>- 0x8 (SIZE_09_BIT): 9-bit Control Word</li> <li>- 0x9 (SIZE_10_BIT): 10-bit Control Word</li> <li>- 0xa (SIZE_11_BIT): 11-bit Control Word</li> <li>- 0xb (SIZE_12_BIT): 12-bit Control Word</li> <li>- 0xc (SIZE_13_BIT): 13-bit Control Word</li> <li>- 0xd (SIZE_14_BIT): 14-bit Control Word</li> <li>- 0xe (SIZE_15_BIT): 15-bit Control Word</li> <li>- 0xf (SIZE_16_BIT): 16-bit Control Word</li> </ul>	0
SRL	[11]	R/W	<p>Shift Register Loop.</p> <p>Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. Can be used in both serial-slave and serial-master modes.</p> <p>When the DW_apb_ssi is configured as a slave in loopback mode, the ss_in_n and ssi_clk signals must be provided by an external source. In this mode, the slave cannot generate these signals because there is nothing to which to loop back</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (TESTING_MODE): Test mode: Tx &amp; Rx shift reg connected</li> <li>- 0x0 (NORMAL_MODE): Normal mode operation</li> </ul>	0

SLV_OE	[10]	R/W	<p>Slave Output Enable. Relevant only when the DW_apb_ssi is configured as a serial-slave device. When configured as a serial master, this bit field has no functionality. This bit enables or disables the setting of the ssi_oe_n output from the DW_apb_ssi serial slave. When SLV_OE = 1, the ssi_oe_n output can never be active. When the ssi_oe_n output controls the tri-state buffer on the txd output from the slave, a high impedance state is always present on the slave txd output when SLV_OE = 1.</p> <p>This is useful when the master transmits in broadcast mode (master transmits data to all slave devices). Only one slave may respond with data on the master rxd line. This bit is enabled after reset and must be disabled by software (when broadcast mode is used), if you do not want this device to respond with data.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x1 (DISABLED): Slave Output is disabled</li> <li>- 0x0 (ENABLED): Slave Output is enabled</li> </ul>	0
TMOD	[9:8]	R/W	<p>Transfer Mode.</p> <p>Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid.</p> <p>In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer.</p> <p>In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer.</p> <p>In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor.</p> <p>In eeprom-read mode, receive data is not valid while control data is being transmitted. When all control data is sent to the EEPROM, receive data becomes valid and transmit data becomes invalid. All data in the transmit FIFO is considered control data in this mode. This transfer mode is only valid when the DW_apb_ssi is configured as master device.</p> <p>00 - Transmit &amp; Receive 01 - Transmit Only 10 - Receive Only 11 - EEPROM Read</p> <p>When SSI_SPI_MODE is either set to "Dual" or "Quad" or "Octal" mode and SPI_FRF is not set to 2'b00. There are only two valid combinations:</p> <p>10 - Read 01 - Write</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (TX_AND_RX): Transmit &amp; receive</li> <li>- 0x1 (TX_ONLY): Transmit only mode or Write (SPI_FRF != 2'b00)</li> <li>- 0x2 (RX_ONLY): Receive only mode or Read (SPI_FRF != 2'b00)</li> <li>- 0x3 (EEPROM_READ): EEPROM Read mode</li> </ul>	0
SCPOL	[7]	R/W	<p>Serial Clock Polarity.</p> <p>Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the DW_apb_ssi master is not actively transferring data on the serial bus.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (SCLK_LOW): Inactive state of serial clock is low</li> <li>- 0x1 (SCLK_HIGH): Inactive state of serial clock is high</li> </ul>	0
SCPH	[6]	R/W	<p>Serial Clock Phase.</p> <p>Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal.</p> <p>When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (SCPH_MIDDLE): Serial clock toggles in middle of first data bit</li> <li>- 0x1 (SCPH_START): Serial clock toggles at start of first data bit</li> </ul>	0
FRF	[5:4]	R/W	<p>Frame Format. Selects which serial protocol transfers the data.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (MOTOROLA_SPI): Motorola SPI Frame Format</li> <li>- 0x1 (TEXAS_SSP): Texas Instruments SSP Frame Format</li> <li>- 0x2 (NS_MICROWIRE): National Microwire Frame Format</li> <li>- 0x3 (RESERVED): Reserved value</li> </ul>	0

DFS	[3:0]	R	<p>Data Frame Size.  This register field is only valid when SSI_MAX_XFER_SIZE is configured to 16. If SSI_MAX_XFER_SIZE is configured to 32, then writing to this field will not have any effect.</p> <p>Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded.</p> <p>You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data.</p> <p>Note: When SSI_SPI_MODE is either set to "Dual" or "Quad" or "Octal" mode and SPI_FRF is not set to 2'b00.</p> <ul style="list-style-type: none"> <li>- DFS value should be multiple of 2 if SPI_FRF = 01,</li> <li>- DFS value should be multiple of 4 if SPI_FRF = 10,</li> <li>- DFS value should be multiple of 8 if SPI_FRF = 11.</li> </ul> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x3 (FRAME_04BITS): 4-bit serial data transfer</li> <li>- 0x4 (FRAME_05BITS): 5-bit serial data transfer</li> <li>- 0x5 (FRAME_06BITS): 6-bit serial data transfer</li> <li>- 0x6 (FRAME_07BITS): 7-bit serial data transfer</li> <li>- 0x7 (FRAME_08BITS): 8-bit serial data transfer</li> <li>- 0x8 (FRAME_09BITS): 9-bit serial data transfer</li> <li>- 0x9 (FRAME_10BITS): 10-bit serial data transfer</li> <li>- 0xa (FRAME_11BITS): 11-bit serial data transfer</li> <li>- 0xb (FRAME_12BITS): 12-bit serial data transfer</li> <li>- 0xc (FRAME_13BITS): 13-bit serial data transfer</li> <li>- 0xd (FRAME_14BITS): 14-bit serial data transfer</li> <li>- 0xe (FRAME_15BITS): 15-bit serial data transfer</li> <li>- 0xf (FRAME_16BITS): 16-bit serial data transfer</li> </ul>	0
-----	-------	---	---	---

### 19.3.3 CTRLR1

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
NDF	[15:0]	R/W	<p>Number of Data Frames.</p> <p>When TMOD = 10 or TMOD = 11 , this register field sets the number of data frames to be continuously received by the DW_apb_ssi. The DW_apb_ssi continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer.</p> <p>When the DW_apb_ssi is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the DW_apb_ssi is configured as a serial slave.</p>	0

### 19.3.4 SSIENR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SSI_EN	[0]	R/W	<p>SSI Enable. Enables and disables all DW_apb_ssi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the DW_apb_ssi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.</p> <p>Values: - 0x0 (DISABLE): Disables Serial Transfer - 0x1 (ENABLED): Enables Serial Transfer</p>	0

### 19.3.5 MWCR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
MHS	[2]	R/W	<p>Microwire Handshaking. Relevant only when the DW_apb_ssi is configured as a serial-master device. When configured as a serial slave, this bit field has no functionality. Used to enable and disable the busy/ready handshaking interface for the Microwire protocol. When enabled, the DW_apb_ssi checks for a ready status from the target slave, after the transfer of the last data/control bit, before clearing the BUSY status in the SR register.</p> <p>Values: - 0x0 (DISABLE): Handshaking interface is disabled - 0x1 (ENABLED): Handshaking interface is enabled</p>	0

MDD	[1]	R/W	<p>Microwire Control. Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the DW_apb_ssi MacroCell from the external serial device. When this bit is set to 1, the data word is transmitted from the DW_apb_ssi MacroCell to the external serial device.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (RECEIVE): SSI receives data</li> <li>- 0x1 (TRANSMIT): SSI transmits data</li> </ul>	0
MWMOD	[0]	R/W	<p>Microwire Transfer Mode. Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (NON_SEQUENTIAL): Non-Sequential Microwire Transfer</li> <li>- 0x1 (SEQUENTIAL): Sequential Microwire Transfer</li> </ul>	0

### 19.3.6 SER

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SER	[0]	R/W	<p>Slave Select Enable Flag. Each bit in this register corresponds to a slave select line (ss_0_n) from the DW_apb_ssi master. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (NOT_SELECTED): No slave selected</li> <li>- 0x1 (SELECTED): Slave is selected</li> </ul>	0

### 19.3.7 BAUDR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x14 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
SCKDV	[15:0]	R/W	<p>SSI Clock Divider.</p> <p>The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation:</p> $Fsclk\_out = Fssi\_clk/SCKDV$ <p>where SCKDV is any even value between 2 and 65534. For example:</p> <p>for <math>Fssi\_clk = 3.6864\text{MHz}</math> and <math>SCKDV = 2</math> <math>Fsclk\_out = 3.6864/2 = 1.8432\text{MHz}</math></p>	0

### 19.3.8 TXFTLR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	Reserved	-
TFT	[4:0]	R/W	<p>Transmit FIFO Threshold.</p> <p>Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. The FIFO depth is configurable in the range 2-256; this register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than or equal to the depth of the FIFO, this field is not written and retains its current value. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered. For information on the Transmit FIFO Threshold values, see the "Master SPI and SSP Serial Transfers" in the DW_apb_ssi Databook.</p> <p>ssi_txe_intr is asserted when TFT or less data entries are present in transmit FIFO</p> <p>Range Variable[x]: Master : 4, Slave: 2</p>	0

### 19.3.9 RXFTLR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x1C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	Reserved	-

RFT	[4:0]	R/W	<p>Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt. The FIFO depth is configurable in the range 2-256. This register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than the depth of the FIFO, this field is not written and retains its current value. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered. For information on the Receive FIFO Threshold values, see the "Master SPI and SSP Serial Transfers" in the DW_apb_ssi Databook. ssi_rxf_intr is asserted when RFT or more data entries are present in receive FIFO. Range Variable[x]: Master : 4, Slave: 2</p>	0
-----	-------	-----	--	---

### 19.3.10 TXFLR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
TXTFL	[5:0]	R/W	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO. Range Variable[x]: Master : 5, Slave: 3	0

### 19.3.11 RXFLR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x24 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
RXTFL	[5:0]	R/W	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. Range Variable[x]: Master : 5, Slave: 3	0

### 19.3.12 SR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x28 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:7]	-	Reserved	-
DCOL	[6]	R	Data Collision Error. Relevant only when the DW_apb_ssi is configured as a master device. This bit will be set if ss_in_n input is asserted by other master, when the DW_apb_ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. Values: - 0x0 (NO_ERROR_CONDITION): No Error - 0x1 (TX_COLLISION_ERROR): Transmit Data Collision Error	0
TXE	[5]	R	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the DW_apb_ssi is configured as a slave device. Data from the previous transmission is resent on the txd line. This bit is cleared when read. Values: - 0x0 (NO_ERROR): No Error - 0x1 (TX_ERROR): Transmission Error	0
RFF	[4]	R	Data Collision Error. Relevant only when the DW_apb_ssi is configured as a master device. This bit will be set if ss_in_n input is asserted by other master, when the DW_apb_ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. Values: - 0x0 (NO_ERROR_CONDITION): No Error - 0x1 (TX_COLLISION_ERROR): Transmit Data Collision Error	0
RFNE	[3]	R	Data Collision Error. Relevant only when the DW_apb_ssi is configured as a master device. This bit will be set if ss_in_n input is asserted by other master, when the DW_apb_ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. Values: - 0x0 (NO_ERROR_CONDITION): No Error - 0x1 (TX_COLLISION_ERROR): Transmit Data Collision Error	0

TFE	[2]	R	<p>Data Collision Error. Relevant only when the DW_apb_ssi is configured as a master device. This bit will be set if ss_in_n input is asserted by other master, when the DW_apb_ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (NO_ERROR_CONDITION): No Error</li> <li>- 0x1 (TX_COLLISION_ERROR): Transmit Data Collision Error</li> </ul>	0
TFNF	[1]	R	<p>Data Collision Error. Relevant only when the DW_apb_ssi is configured as a master device. This bit will be set if ss_in_n input is asserted by other master, when the DW_apb_ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (NO_ERROR_CONDITION): No Error</li> <li>- 0x1 (TX_COLLISION_ERROR): Transmit Data Collision Error</li> </ul>	0
BUSY	[0]	R	<p>Data Collision Error. Relevant only when the DW_apb_ssi is configured as a master device. This bit will be set if ss_in_n input is asserted by other master, when the DW_apb_ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (NO_ERROR_CONDITION): No Error</li> <li>- 0x1 (TX_COLLISION_ERROR): Transmit Data Collision Error</li> </ul>	0

### 19.3.13 IMR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x2C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
MSTIM	[5]	R/W	<p>Multi-Master Contention Interrupt Mask. This bit field is not present if the DW_apb_ssi is configured as a serial-slave device.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (MASKED): ssi_mst_intr interrupt is masked</li> <li>- 0x1 (UNMASKED): ssi_mst_intr interrupt is not masked</li> </ul>	0
RXFIM	[4]	R/W	<p>Receive FIFO Full Interrupt Mask</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (MASKED): ssi_rxf_intr interrupt is masked</li> <li>- 0x1 (UNMASKED): ssi_rxf_intr interrupt is not masked</li> </ul>	0
RXOIM	[3]	R/W	<p>Receive FIFO Overflow Interrupt Mask</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (MASKED): ssi_rxo_intr interrupt is masked</li> <li>- 0x1 (UNMASKED): ssi_rxo_intr interrupt is not masked</li> </ul>	0
RXUIM	[2]	R/W	<p>Receive FIFO Underflow Interrupt Mask</p> <p>Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (MASKED): ssi_rxu_intr interrupt is masked</li> <li>- 0x1 (UNMASKED): ssi_rxu_intr interrupt is not masked</li> </ul>	0

TXOIM	[1]	R/W	Transmit FIFO Overflow Interrupt Mask Values: - 0x0 (MASKED): ssi_txo_intr interrupt is masked - 0x1 (UNMASKED): ssi_txo_intr interrupt is not masked	0
TXEIM	[0]	R/W	Transmit FIFO Empty Interrupt Mask Values: - 0x0 (MASKED): ssi_txe_intr interrupt is masked - 0x1 (UNMASKED): ssi_txe_intr interrupt is not masked	0

### 19.3.14 ISR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
MSTIS	[5]	R	Multi-Master Contention Interrupt Status. This bit field is not present if the DW_apb_ssi is configured as a serial-slave device. Values: - 0x0 (INACTIVE): ssi_mst_intr interrupt not active after masking - 0x1 (ACTIVE): ssi_mst_intr interrupt is active after masking	0
RXFIS	[4]	R	Receive FIFO Full Interrupt Status Values: - 0x0 (INACTIVE): ssi_rxf_intr interrupt is not active after masking - 0x1 (ACTIVE): ssi_rxf_intr interrupt is full after masking	0
RXOIS	[3]	R	Receive FIFO Overflow Interrupt Status Values: - 0x0 (INACTIVE): ssi_rxo_intr interrupt is not active after masking - 0x1 (ACTIVE): ssi_rxo_intr interrupt is active after masking	0
RXUIS	[2]	R	Receive FIFO Underflow Interrupt Status Values: - 0x0 (INACTIVE): ssi_rxu_intr interrupt is not active after masking - 0x1 (ACTIVE): ssi_rxu_intr interrupt is active after masking	0
TXOIS	[1]	R	Transmit FIFO Overflow Interrupt Status Values: - 0x0 (INACTIVE): ssi_txo_intr interrupt is not active after masking - 0x1 (ACTIVE): ssi_txo_intr interrupt is active after masking	0
TXEIS	[0]	R	Transmit FIFO Empty Interrupt Status Values: - 0x0 (INACTIVE): ssi_txe_intr interrupt is not active after masking - 0x1 (ACTIVE): ssi_txe_intr interrupt is active after masking	0

### 19.3.15 RISR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x34 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
MSTIR	[5]	R	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the DW_apb_ssi is configured as a serial-slave device. Values: - 0x0 (INACTIVE): ssi_mst_intr interrupt is not active prior to masking - 0x1 (ACTIVE): ssi_mst_intr interrupt is active prior masking	0
RXFIR	[4]	R	Receive FIFO Full Raw Interrupt Status Values: - 0x0 (INACTIVE): ssi_rxf_intr interrupt is not active prior to masking - 0x1 (ACTIVE): ssi_rxf_intr interrupt is active prior to masking	0
RXOIR	[3]	R	Receive FIFO Overflow Raw Interrupt Status Values: - 0x1 (ACTIVE): ssi_rxo_intr interrupt is not active prior to masking - 0x0 (INACTIVE): ssi_rxo_intr interrupt is active prior masking	0
RXUIR	[2]	R	Receive FIFO Underflow Raw Interrupt Status Values: - 0x0 (INACTIVE): ssi_rxu_intr interrupt is not active prior to masking - 0x1 (ACTIVE): ssi_rxu_intr interrupt is active prior to masking	0
TXOIR	[1]	R	Transmit FIFO Overflow Raw Interrupt Status Values: - 0x0 (INACTIVE): ssi_txo_intr interrupt is not active prior to masking - 0x1 (ACTIVE): ssi_txo_intr interrupt is active prior masking	0
TXEIR	[0]	R	Transmit FIFO Empty Raw Interrupt Status Values: - 0x0 (INACTIVE): ssi_txe_intr interrupt is not active prior to masking - 0x1 (ACTIVE): ssi_txe_intr interrupt is active prior masking	0

### 19.3.16 TXOICR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)

- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x38 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
TXOICR	[0]	R	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.	0

### 19.3.17 RXOICR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x3C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
RXOICR	[0]	R	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.	0

### 19.3.18 RXUICR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x40 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value

RSVD	[31:1]	-	Reserved	-
RXUICR	[0]	R	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect.	0

### 19.3.19 MSTICR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x44 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
MSTICR	[0]	R	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect.	0

### 19.3.20 ICR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x48 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
ICR	[0]	R	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.	0

### 19.3.21 DMACR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x4C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
RDMAE	[0]	R/W	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel Values: - 0x0 (DISABLE): Receive DMA disabled - 0x1 (ENABLED): Receive DMA enabled	0
TDMAE	[0]	R/W	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. Values: - 0x0 (DISABLE): Transmit DMA disabled - 0x1 (ENABLED): Transmit DMA enabled	0

### 19.3.22 DMATDLR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x50 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	Reserved	-
DMATDL	[4:0]	R/W	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1. For information on the DMATDL decode values, see the "Slave SPI and SSP Serial Transfers" section in the DW_app_ssi Databook. dma_tx_req is asserted when DMATDL or less data entries are present in the transmit FIFO Range Variable[x]: Master : 4, Slave: 2	0

### 19.3.23 DMARDLR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x54 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	Reserved	-
DMARDL	[4:0]	R/W	<p>Receive Data Level.</p> <p>This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1. For information on the DMARDL decode values, see the "Slave SPI and SSP Serial Transfers" section in the DW_apb_ssi Databook.</p> <p>dma_rx_req is asserted when DMARDL or more valid data entries are present in the receive FIFO.</p> <p>Range Variable[x]: Master : 4, Slave: 2</p>	0

### 19.3.24 IDR

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x58 Reset Value = Undefined

Name	Bit	Type	Description	Reset Value
IDCODE	[31:0]	R	<p>Identification code.</p> <p>The register contains the peripheral's identification code, which is written into the register at configuration time using CoreConsultant.</p>	:no reset

### 19.3.25 SSI\_VERSION\_ID

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x5C Reset Value = Undefined

Name	Bit	Type	Description	Reset Value
SSI_COMP_VERSION	[31:0]	R	Contains the hex representation of the Synopsys component version. Consists of ASCII value for each number in the version, followed by *. For example 32_30_31_2A represents the version 2.01*.	:no reset

### 19.3.26 DRx

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0x60 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
DR	[31:0]	R/W	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. If SSI_MAX_XFER_SIZE configuration parameter is set to 32, all 32 bits are valid. Otherwise, only 16 bits ([15:0]) of the register are valid. Read = Receive FIFO buffer Write = Transmit FIFO buffer.	0

### 19.3.27 RX\_SAMPLE\_DLY

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0xF0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
RSD	[7:0]	R/W	Rxd Sample Delay. This register is used to delay the sample of the rxd input port. Each value represents a single ssi_clk delay on the sample of rxd. Note: If this register is programmed with a value that exceeds the depth of the internal shift registers (SSI_RX_DLY_SR_DEPTH) zero delay will be applied to the rxd sample.	0

### 19.3.28 SPI\_CTRLR0

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0xF4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:19]	-	Reserved	-
SPI_RXDS_EN	[18]	R	Read data strobe enable bit. Once this bit is set to 1 DW_apb_ssi will use Read data strobe (rxds) to capture read data in DDR mode.	0
INST_DDR_EN	[17]	R/W	Instruction DDR Enable bit. This will enable Dual-data rate transfer for Instruction phase. SPI0 : Support DDR, SPI1/SPI2 : No support	0
SPI_DDR_EN	[16]	R/W	SPI DDR Enable bit. This will enable Dual-data rate transfers in Dual/Quad/Octal frame formats of SPI. SPI0 : Support DDR, SPI1/SPI2 : No support	0
WAIT_CYCLES	[15:11]	R/W	Wait cycles Number of wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. This value is specified as number of SPI clock cycles. For information on the WAIT_CYCLES decode value, see "Read Operation in Enhanced SPI Modes" section in the DW_apb_ssi Databook.	0

RSVD	[10]	-	Reserved	-				
INST_L	[9:8]	R/W	<p>Instruction Length Dual/Quad/Octal mode instruction length in bits. Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (INST_L_0): 0-bit (No Instruction)</li> <li>- 0x1 (INST_L_1): 4-bit Instruction</li> <li>- 0x2 (INST_L_2): 8-bit Instruction</li> <li>- 0x3 (INST_L_3): 16-bit Instruction</li> </ul>	0				
RSVD	[7:6]	-	Reserved	-				
ADDR_L	[5:2]	R/W	<p>Address Length. This bit defines Length of Address to be transmitted. Only after this much bits are programmed in to the FIFO the transfer can begin. For information on the ADDR_L_decode value, see "Read Operation in Enhanced SPI Modes" section in the DW_apb_ssi Databook. Values:</p> <ul style="list-style-type: none"> <li>- 0x0 (ADDR_L_0): 0-bit Address Width</li> <li>- 0x1 (ADDR_L_1): 4-bit Address Width</li> <li>- 0x2 (ADDR_L_2): 8-bit Address Width</li> <li>- 0x3 (ADDR_L_3): 12-bit Address Width</li> <li>- 0x4 (ADDR_L_4): 16-bit Address Width</li> <li>- 0x5 (ADDR_L_5): 20-bit Address Width</li> <li>- 0x6 (ADDR_L_6): 24-bit Address Width</li> <li>- 0x7 (ADDR_L_7): 28-bit Address Width</li> <li>- 0x8 (ADDR_L_8): 32-bit Address Width</li> <li>- 0x9 (ADDR_L_9): 36-bit Address Width</li> <li>- 0xa (ADDR_L_10): 40-bit Address Width</li> <li>- 0xb (ADDR_L_11): 44-bit Address Width</li> <li>- 0xc (ADDR_L_12): 48-bit Address Width</li> <li>- 0xd (ADDR_L_13): 52-bit Address Width</li> <li>- 0xe (ADDR_L_14): 56-bit Address Width</li> <li>- 0xf (ADDR_L_15): 60-bit Address Width</li> </ul>	0				
TRANS_TYPE	[1:0]	R/W	<p>Address and instruction transfer format. Selects whether DW_apb_ssi will transmit instruction/address either in Standard SPI mode or the SPI mode selected in CTRLR0.SPI_FRF field.</p> <table border="0" style="width: 100%;"> <tr> <td>00 - Instruction and Address will be sent in Standard SPI Mode.</td> </tr> <tr> <td>01 - Instruction will be sent in Standard SPI Mode and Address will be sent in the mode specified by CTRLR0.SPI_FRF.</td> </tr> <tr> <td>10 - Both Instruction and Address will be sent in the mode specified by SPI_FRF.</td> </tr> <tr> <td>11 - Reserved.</td> </tr> </table>	00 - Instruction and Address will be sent in Standard SPI Mode.	01 - Instruction will be sent in Standard SPI Mode and Address will be sent in the mode specified by CTRLR0.SPI_FRF.	10 - Both Instruction and Address will be sent in the mode specified by SPI_FRF.	11 - Reserved.	0
00 - Instruction and Address will be sent in Standard SPI Mode.								
01 - Instruction will be sent in Standard SPI Mode and Address will be sent in the mode specified by CTRLR0.SPI_FRF.								
10 - Both Instruction and Address will be sent in the mode specified by SPI_FRF.								
11 - Reserved.								

### 19.3.29 TXD\_DRIVE\_EDGE

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0xF8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
TDE	[7:0]	R/W	TXD Drive edge - value of which decides the driving edge of transmit data. The maximum value of this register is = (BAUDR/2) -1.	0

### 19.3.30 RSVD

- Base Address: 20800000(SPI0)
- Base Address: 20810000(SPI1)
- Base Address: 20820000(SPI2)
- Base Address: 20830000(QSPI0)
- Base Address: 20840000(QSPI1)
- Base Address: 20850000(QSPI2)
- Base Address: 20860000(QSPI3)
- Base Address: 20870000(QSPI4)
- Address = Base Address + 0xFC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:0]	-	Reserved	-

# 20 UART/USART

## 20.1 Features

- 9-bit serial data support
- False start bit detection
- Programmable fractional baud rate support
- Multi-drop RS485 interface support
- Configurable parameters for the following:
  - APB data bus widths of 8, 16 and 32
  - Additional DMA interface signals for compatibility with DesignWare DMA interface
  - DMA interface signal polarity
  - Transmit and receive FIFO depths of 0, 16, 32, 64, 128, 256, 512, 1024, 2048
  - Internal or external FIFO (RAM) selection
  - Use of two clocks—pclk and sclk—instead of just pclk
  - IrDA 1.0 SIR mode support with up to 115.2 Kbaud data rate and a pulse duration (width) as specified in the IrDA physical layer specification: width = 3/16  $\mu$ bit period
  - IrDA 1.0 SIR low-power reception capabilities
  - Baud clock reference output signal
  - Clock gate enable output(s) used to indicate that the TX and RX pipeline is clear (no data) and no activity has occurred for more than one character time, so that clocks can be gated
  - FIFO access mode—for FIFO testing—enabling the master to write to the receive FIFO and read from the transmit FIFO
  - Additional FIFO status registers
  - Shadow registers to reduce software overhead and also include a software programmable reset
  - Auto Flow Control mode, as specified in the 16750 standard
  - Loopback mode that enables greater testing of Modem Control and Auto Flow Control features (Loopback support in IrDA SIR mode is available)
  - Transmitter Holding Register Empty (THRE) interrupt mode
  - Busy functionality
- Ability to set some configuration parameters during instantiation
- Configuration identification registers present
- Functionality based on the 16550 industry standard
  - Programmable character properties, such as:
    - \* Number of data bits per character (5-8)
    - \* Optional parity bit (with odd, even select or Stick Parity)
    - \* Number of stop bits (1, 1.5 or 2)
  - Line break generation and detection
  - DMA signaling with two programmable modes

- Prioritized interrupt identification
- Programmable FIFO enable/disable
- Programmable serial data baud rate as calculated by the following: baud rate = (serial clock frequency)/(16 $\times$  divisor)
- External read enable signal for RAM wake-up when using external RAMs
- Modem and status lines are independently controlled
- Separate system resets for each clock domain to prevent metastability

## 20.2 Block Diagram

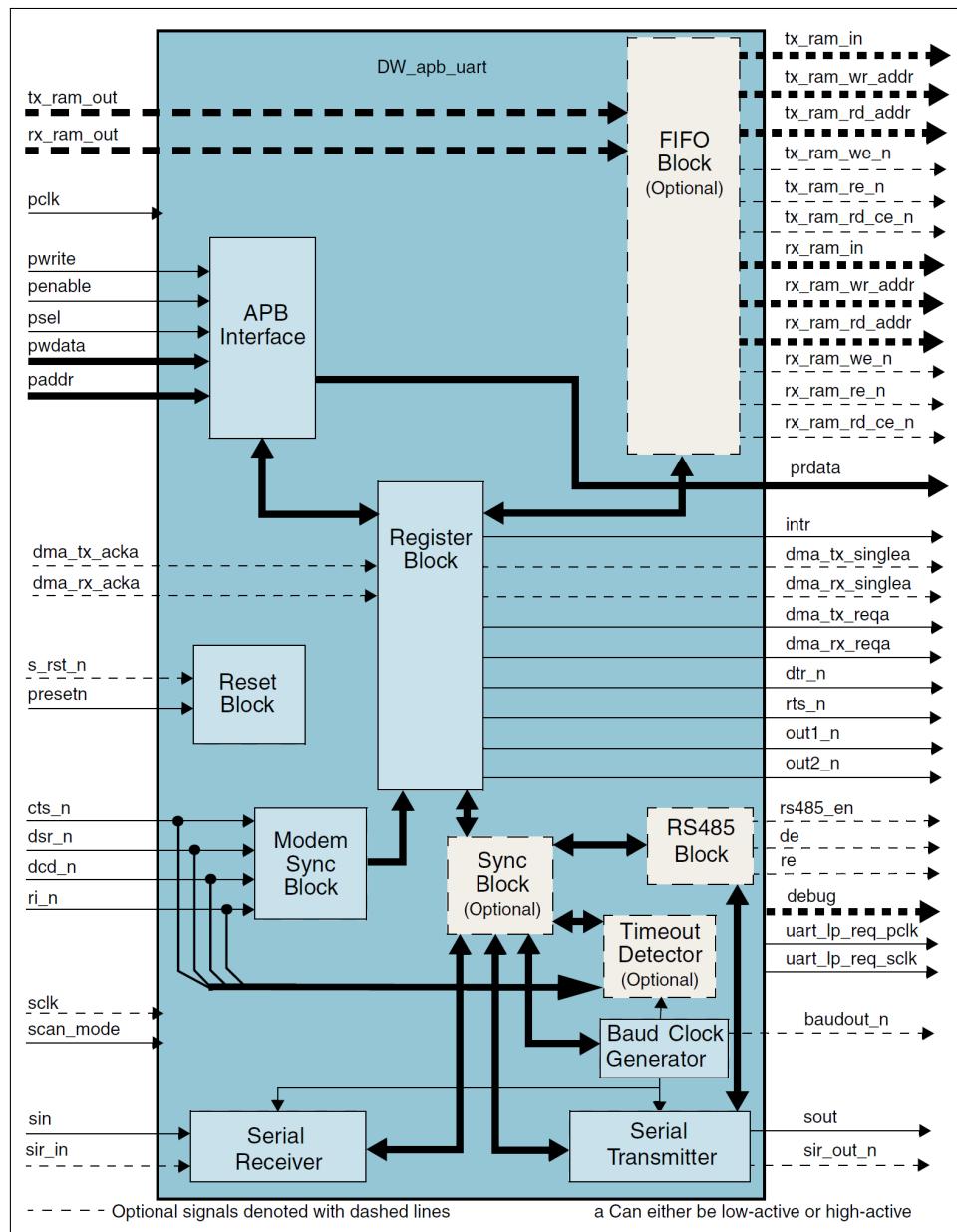


Figure 20.1: UART Functional Block Diagram

The following list describes each of the major blocks shown in Figure 20.1:

- Reset block – resets clock domains.
- APB slave interface – connects to APB bus.
- Register block – responsible for the main UART functionality including control, status and interrupt generation.
- Modem Synchronization block – synchronizes the modem input signal.

- FIFO block (optional) – responsible for FIFO control and storage—when using internal RAM—or optionally signaling to control external RAM.
- Synchronization block (optional) – implemented when the peripheral is configured to have a separate serial data clock (i.e. two clock implementation).
- Timeout Detector block (optional) – indicates the absence of character data movement in the receiver FIFO within a given time period; this is used to generate character timeout interrupts when enabled. This block can also have optional clock gate enable outputs—uart\_lp\_req\_pclk for single clock implementations or uart\_lp\_req\_pclk and uart\_lp\_req\_sclk for two clock implementations—in order to indicate: TX and RX pipeline is clear; that is, there is no data
  - No activity has occurred
  - Modem control input signals have not changed within a given time period
- Baud Clock Generator – produces the transmitter and receiver baud clock along with the output reference clock signal (baudout\_n).
- Serial Transmitter – converts the parallel data—written to the UART—into serial form and adds all additional bits, as specified by the control register, for transmission. These serial data, referred to as a character, can exit the block in two formats:
  - Serial UART
  - IrDA 1.0 SIR
- Serial Receiver – converts the serial data character—specified by the control register—received in either the UART or IrDA 1.0 SIR format to parallel form. This block controls:
  - Parity error detection
  - Framing error detection
  - Line break detection
- RS485 block (optional) – implemented when the peripheral is configured to have an RS485 interface, responsible for the generation of driver enable (de) and receiver enable (re) signals required by the RS485 Transceiver.

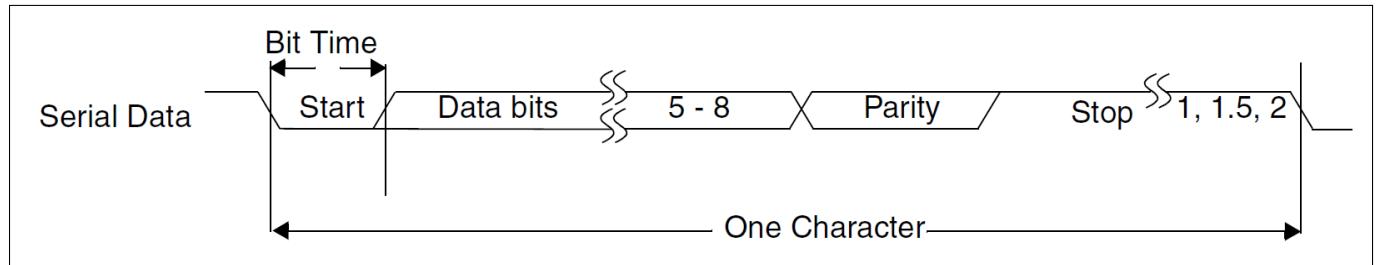


Figure 20.2: Serial Data Format

## 20.3 Functional Description

This chapter describes the functional operation of the UART.

### 20.3.1 UART (RS232) Serial Protocol

Because the serial communication between the UART and a selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data—accompanied by start and stop bits—is referred to as a character, as shown in Figure 20.2.

An additional parity bit can be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure in order to provide the UART with the ability to perform simple error checking on the received data.

The UART Line Control Register (“LCR”) is used to control the serial character characteristics. The individual bits of the data word are sent after the start bit, starting with the least-significant bit (LSB). These are followed by the optional parity bit, followed by the stop bit(s), which can be 1, 1.5, or 2.

#### Note

The STOP bit duration implemented by UART can appear longer due to:

- Idle time inserted between characters for some configurations
- Baud clock divisor values in the transmit direction

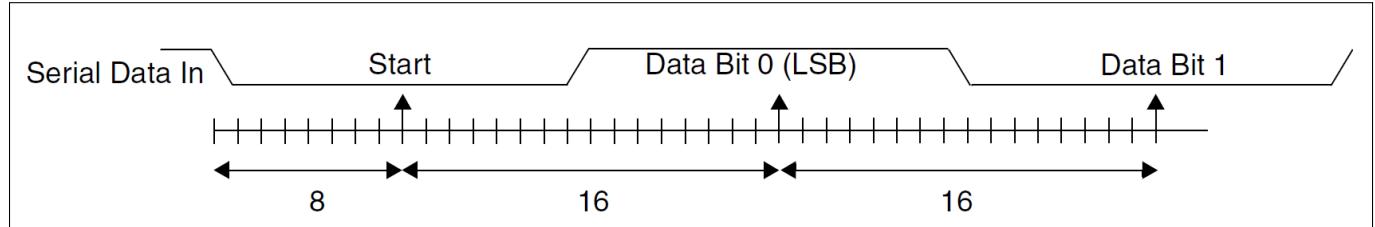
For details on idle time between transmitted transfers, refer to “Back-to-Back Character Stream Transmission”.

All the bits in the transmission are transmitted for exactly the same time duration; the exception to this is the half-stop bit when 1.5 stop bits are used. This duration is referred to as a Bit Period or Bit Time; one Bit Time equals sixteen baud clocks.

To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. Because the exact number of baud clocks is known for which each bit was transmitted, calculating the midpoint for sampling is not difficult; that is, every sixteen baud clocks after the midpoint sample of the start bit.

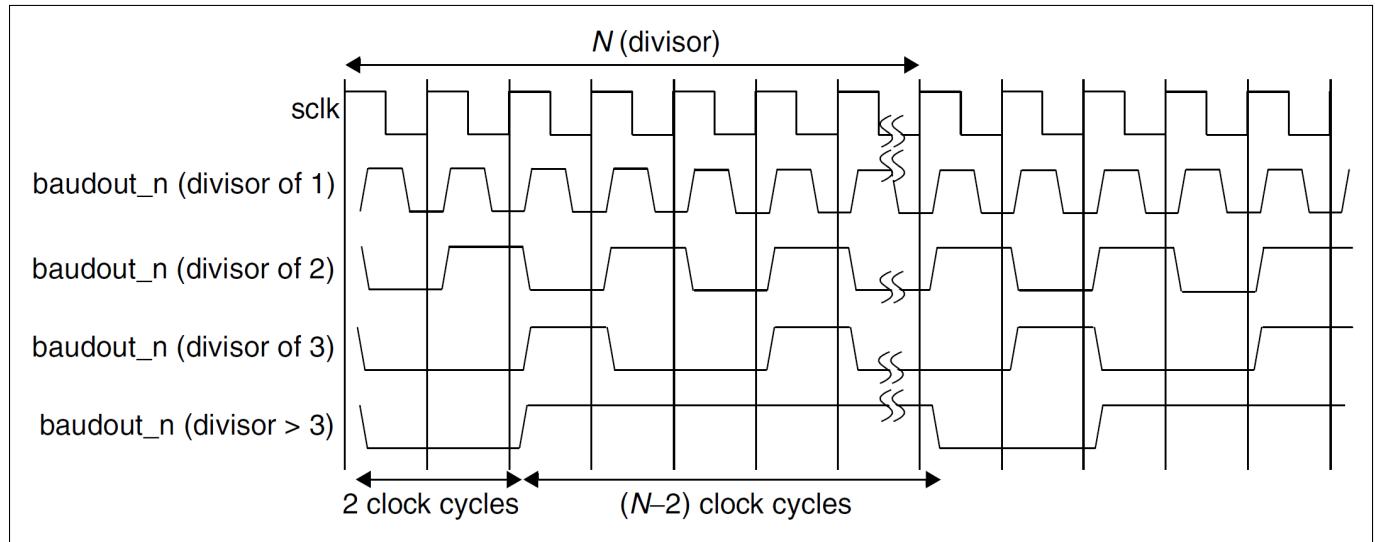
Together with serial input debouncing, this sampling helps to avoid the detection of false start bits. Short glitches are filtered out by debouncing, and no transition is detected on the line. If a glitch is wide enough to avoid filtering

by debouncing, a falling edge is detected. However, a start bit is detected only if the line is again sampled low after half a bit time has elapsed. Figure 20.3 shows the sampling points of the first two bits in a serial character.



**Figure 20.3:** Receiver Serial Data Sample Points

As part of the 16550 standard, an optional baud clock reference output signal (`baudout_n`) provides timing information to receiving devices that require it. The baud rate of the UART is controlled by the serial clock—`sclk` or `pclk` in a single clock implementation—and the Divisor Latch Register (DLH and DLL). Figure 20.4 shows the timing diagram for the `baudout_n` output for different divisor values.



**Figure 20.4:** Baud Clock Reference Timing Diagram

### 20.3.2 9-bit Data Transfer

The UART can be configured to have 9-bit data transfer in both transmit and receive mode. The 9th bit in the character appears after the 8th bit and before the parity bit in the character. Figure 20.5 shows the serial transmission for a character in which D8 represents the 9th bit and also shows general serial transmission in the 9-bit mode.

By enabling 9-bit data transfer mode, UART can be used in multi-drop systems where one master is connected to multiple slaves in a system. The master communicates with one of the slaves. When the master wants to transfer a block of data to a slave, it first sends an address byte to identify the target slave.

The differentiation between the address/data byte is done based on the 9th bit in the incoming character. If the 9th bit is set to 0, then the character represents a data byte. If the 9th bit is set to 1, then the character represents address byte. All the slave systems compare the address byte with their own address and only the target slave (in which the address has matched) is enabled to receive data from the master. The master then starts transmitting

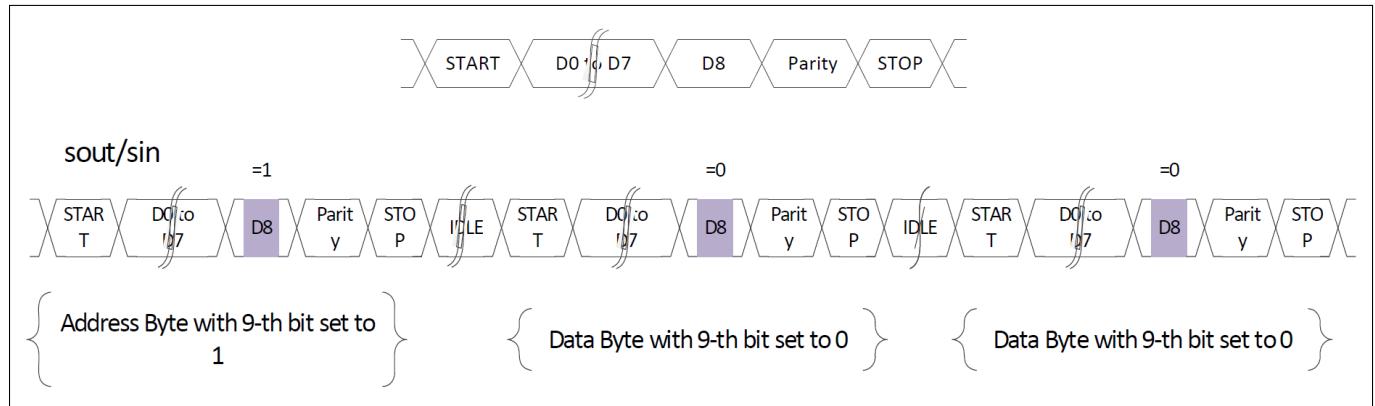


Figure 20.5: 9-Bit Character

data bytes to the target slave. The non-addressed slave systems ignore the incoming data until a new address byte is received.

In Figure 20.5, note that one address is followed by 2 data bytes. The address byte goes out with the 9th bit (D8) set to 1 and the data bytes go out with 9th bit (D8) set to 0. The parity bit is an optional field.

Configuration of the UART for 9-bit data transfer does the following:

- LCR\_EXT[0] bit is used to enable or disable the 9-bit data transfer.
- LCR\_EXT[1] bit is used to choose between hardware and software based address match in the case of receive.
- LCR\_EXT[2] bit is used to enable to send the address in the case of transmit.
- LCR\_EXT[3] bit is used to choose between hardware and software based address transmission.
- TAR and RAR registers are used to transmit address and to match the received address, respectively.
- THR, RBR, STHR and SRBR registers are of 9-bit which is used to do the data transfers in 9-bit mode.
- LSR[8] bit is used to indicate the address received interrupt.

#### Note

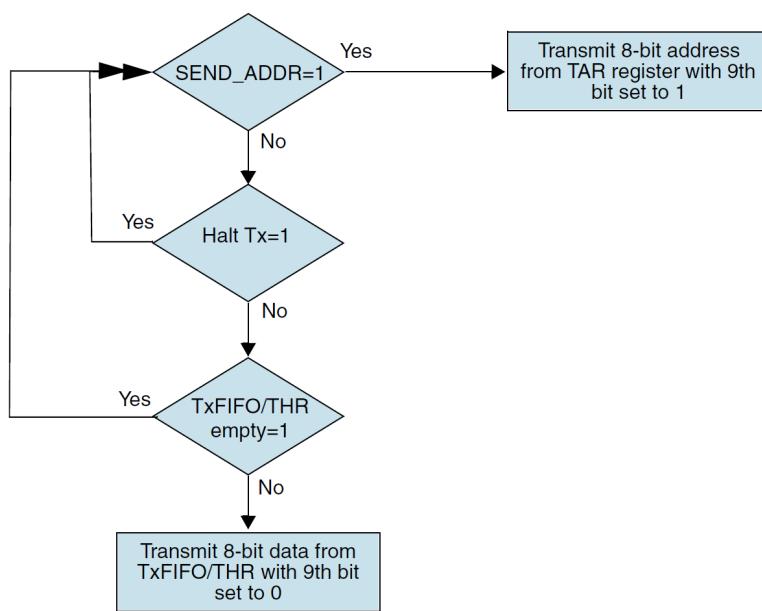
The 9-bit data mode is supported only when the DWC-APB-AdvancedSource source license exists.

##### 20.3.2.1 Transmit Mode

UART supports two types of transmit modes:

- Transmit Mode 0 (when (LCR\_EXT[3]) is set to 0)
- Transmit Mode 1 (when (LCR\_EXT[3]) is set to 1)

**20.3.2.1.1 Transmit Mode 0** In transmit mode 0, the address is programmed in the Transmit Address Register (TAR) register and data is written into the Transmit Holding Register (THR) or the Shadow Transmit Holding Register (STHR). The 9th bit of the THR and STHR register is not applicable in this mode.



**Figure 20.6:** Auto Address Transmit Flow Chart

Figure 20.6 illustrates the transmission of address and data based on SEND\_ADDR (LCR\_EXT[2]), Halt Tx, and TxFIFO/THR empty conditions.

The address of the target slave to which the data is to be transmitted is programmed in the TAR register. You must enable the SEND\_ADDR (LCR\_EXT[2]) bit to transmit the target slave address present in the TAR register on the serial UART line with 9th data bit set to 1 to indicate that the address is being sent to the slave. The UART clears the SEND\_ADDR bit after the address character starts transmitting on the UART line.

The data required to transmit to the target slave is programmed through Transmit Holding Register (THR). The data is transmitted on the UART line with 9th data bit set to 0 to indicate data is being sent to the slave.

If the application is required to fill the data bytes in the TxFIFO before sending the address on the UART line (before setting LCR\_EXT[2]=1), then it is recommended to set the “Halt Tx” to 1 such that UART does not start sending out the data in the TxFIFO as data byte. Once the TxFIFO is filled, then program SEND\_ADDR (LCR\_EXT[2]) to 1 and then set “Halt Tx” to 0.

**20.3.2.1.2 Transmit Mode 1** In transmit mode 1, THR and STHR registers are of 9-bit wide and both address and data are programmed through the THR and STHR registers. The UART does not differentiate between address and data, and both are taken from the TxFIFO. The SEND\_ADDR (LCR\_EXT[2]) bit and Transmit address register (TAR) are not applicable in this mode. The software must pack the 9th bit with 1/0 depending on whether address/data has to be sent.

### 20.3.2.2 Receive Mode

The UART supports two receive modes:

- Hardware Address Match Receive Mode (when ADDR\_MATCH (LCR\_EXT[1]) is set to 1)
- Software Address Match Receive Mode (when ADDR\_MATCH (LCR\_EXT[1]) is set to 0)

**20.3.2.2.1 Hardware Address Match Receive Mode** In the hardware address match receive mode, the UART matches the received character with the address programmed in the Receive Address register (RAR), if the 9th bit of the received character is set to 1. If the received address is matched with the programmed address in RAR register, then subsequent data bytes (with 9th bit set to 0) are pushed into the RxFIFO. If the address matching fails, then UART controller discards further data characters until a matching address is received.

Figure 20.7 illustrates the flow chart for the reception of data bytes based on the address matching feature.

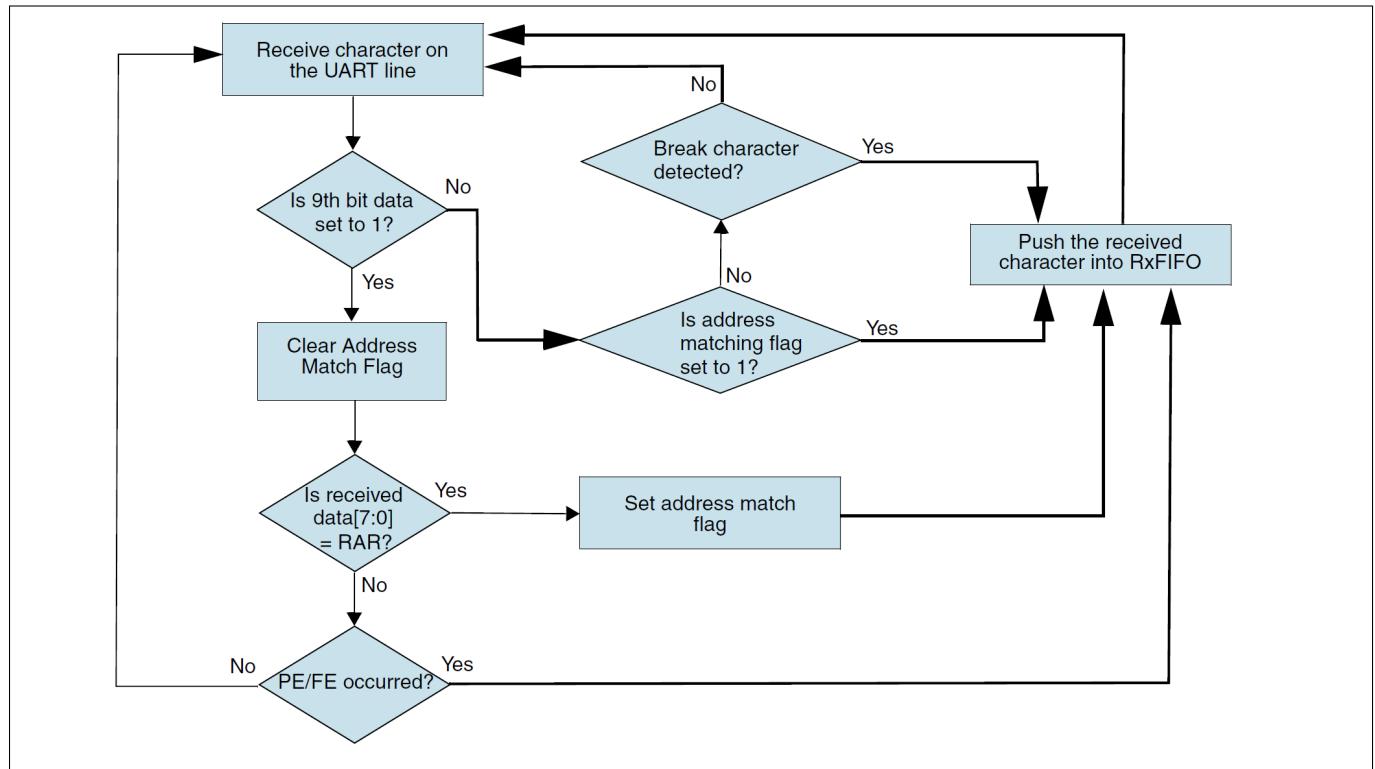


Figure 20.7: Hardware Address Match Receive Mode

UART receives the character irrespective of whether the 9th bit data is set to 1. If 9th bit of the received character is set to 1, then it clears internal address match flag and then compares the received 8-bit character information with the address programmed in the RAR register.

If the received address character matches with the address programmed in the RAR register, then the address match flag is set to 1 and the received character is pushed to the RxFIFO in FIFO-mode or to RBR register in non-FIFO mode and the ADDR\_RCVD bit in LSR register is set to indicate that the address has been received.

In case of parity or if a framing error is found in the received address character and if the address is not matched with the RAR register, then the received address character is still pushed to RxFIFO or RBR register with ADDR\_RCVD and PE/FE error bit set to 1.

The subsequent data bytes (9th bit of received character is set to 0) are pushed to the Rx\_FIFO in FIFO mode or to the RBR register in non-FIFO mode until the new address character is received.

If any break character is received, UART treats it as a special character and pushes to the RxFIFO or RBR register based on the FIFO\_MODE irrespective of address match flag.

Note

The break character can be used to alert the complete system in case all slaves are in sleep mode (entered in to the low power mode). Therefore, the break character is treated as special character.

**20.3.2.2.2 Software Address Match Receive Mode** In this mode of operation, the UART does not perform the address matching for the received address character (9th bit data set to 1) with the RAR register. The UART always receives the 9-bit data and pushes in to RxFIFO in FIFO mode or to the RBR register in non-FIFO mode. The user must compare the address whenever address byte is received and indicated through ADDR\_RCVD bit in the Line Status register. The user can flush/reset the RxFIFO in case of address not matched through 'RCVR FIFO Reset' bit in FIFO control register (FCR).

### 20.3.3 RS485 Serial Protocol

The RS485 standard supports serial communication over a twisted pair configuration, such as RS232. The difference between the RS232 and RS485 standards is its use of a balanced line for transmission. This usage is also known as the differential format that sends the same signal on two separate lines with phase delay and then compares the signals at the end, subtracts any noise, and adds them to regain signal strength. This process allows the RS485 standard to be viable over significantly longer distances than its short range RS232 counterpart.

UART supports the RS485 serial protocol that enables transfer of serial data using the RS485 interface. The driver enable (DE) and receiver enable (RE) signals are generated for enabling the RS485 interface support. The de and re signals are hardware generated and the assertion/de-assertion times for these signals are programmable. The active level of these signals are configurable.

Configuration of the UART for RS485 interface does the following:

1. Bit 0 of the Transceiver Control Register (TCR) enables or disables the RS485 mode.
2. Bit 1 and bit 2 of TCR are used to select the polarity of RE and DE signals.
3. Bit [4:3] of the TCR selects the type of transfer in RS485 mode.
4. Driver output enable (DE\_EN) and Receiver output enable (RE\_EN) registers are used for software control of DE and RE signals.
5. Driver output Enable Timing (DET) register is used to program the assertion and deassertion timings of DE signal.
6. TurnAround Timing (TAT) register is used to program the turnaround time from DE to RE and RE to DE.

Note

RS485 interface mode is supported only when the source license DWC-APB-AdvancedSource exists.

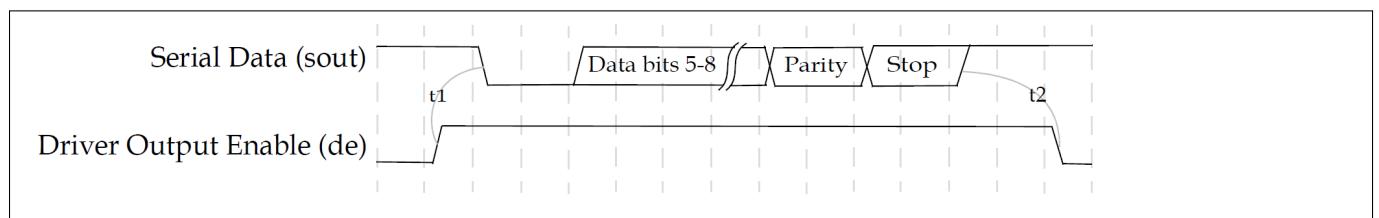
#### 20.3.3.1 DE Assertion and De-assertion Timing

The assertion and deassertion timings of the DE signal are controlled through the DET register:

- DE assertion time (DET[7:0]): The assertion time is the time between the activation of the DE signal and the beginning of the START bit. The value represented is in terms of serial clock cycles.
- DE de-assertion time (DET[15:8]): The de-assertion time is the time between the end of the last stop bit, in a transmitted character, and the de-activation of the DE signal. The value represented is in terms of serial clock cycles.

Hardware ensures that these values are met for DE assertion and DE deassertion before/after active data transmission.

In Figure 20.8, t1 represents DE assertion time and t2 represents DE de-assertion time. Note that for simplicity only one data is illustrated in Figure 20.8; however, DE will not get de-asserted if there are more data characters in transmit FIFO. DE gets de-asserted only after all the data characters are transmitted.



**Figure 20.8: DE Assertion and De-Assertion**

### 20.3.3.2 RS485 Modes

UART consists of the following RS485 modes based on the XFER\_MODE field in the Transceiver Control Register (TCR) register:

- Full Duplex Mode – In this mode, XFER\_MODE of TCR is set to 0.
- Software-Controlled Half Duplex Mode – In this mode, XFER\_MODE of TCR is set to 1.
- Hardware-Controlled Half Duplex Mode – In this mode, XFER\_MODE of TCR is set to 2

#### 20.3.3.2.1 Full Duplex Mode

The full duplex mode supports both transmit and receive transfers simultaneously.

In Full Duplex mode, the de signal:

- Goes active if both these conditions are satisfied:
  - When the DE Enable (DE\_EN[0]) field of Driver Output Enable Register is set to 1.
  - Transmitter Holding Register is not empty in non-FIFO mode or transmitter FIFO is not empty in FIFO mode.
- Goes inactive if both these conditions are satisfied
  - When the current ongoing transmitting serial transfer is completed.
  - Either DE Enable (DE\_EN[0]) of Driver Output Enable Register is set to 0, transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in non-FIFO mode.

In Full Duplex mode, the re signal:

- Goes active when RE Enable (RE\_EN[0]) of Receiver Output Enable Register is set to 1.
- Goes inactive when RE Enable (RE\_EN[0]) of Receiver Output Enable Register is set to 0.

The user can choose when to transmit or when to receive. Both 're' and 'de' can be simultaneously asserted or de-asserted at any time. UART does not impose any turnaround time between transmit and receive ('de to re') or receive to transmit ('re to de') in this mode. This mode can directly be used in full duplex operation where separate differential pair of wires is present for transmit and receive.

**20.3.3.2.2 Software-Controlled Half Duplex Mode** The software-controlled half duplex mode supports either transmit or receive transfers at a time but not both simultaneously. The switching between transmit to receive or receive to transmit is through programming the Driver output enable (DE\_EN) and Receiver output enable (RE\_EN) registers.

In software-controlled Half Duplex mode, the de signal:

- Goes active if the following conditions are satisfied:
  - The DE Enable (DE\_EN[0]) field of the Driver Output Enable Register is set to 1.
  - Transmitter Holding Register is not empty in non-FIFO mode or transmitter FIFO is not empty in FIFO mode.
  - If any receive transfer is ongoing, then the signal waits until receive has finished, and after the turnaround time counter ('re to de') has elapsed.
- Goes inactive if the following conditions are satisfied:
  - The current ongoing transmitting serial transfer is completed.
  - The DE Enable (DE\_EN[0]) field of Driver Output Enable Register is set to 0.
  - Either transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in nonFIFO mode.

In software-controlled half duplex mode, the re signal:

- Goes active if the following conditions are satisfied:
  - When RE Enable (RE\_EN[0]) field of Receiver Output Enable Register is set to 1.
  - If any transmit transfer is ongoing, then the signal waits until transmit has finished and after the turnaround time counter ('de to re') has elapsed.
- Goes in-active under the following conditions:
  - The current ongoing receive serial transfer is completed.
  - When RE Enable (RE\_EN[0]) of Receiver Output Enable Register is set to 0.

The user must enable either DE or RE but not both at any point of time. As 're' and 'de' signals are mutually exclusive, the user must ensure that both of them are not programmed to be active at any point of time. In this mode, the hardware ensures that a proper turnaround time is maintained while switching from 're' to 'de' or from 'de' to 're' (value of turnaround is obtained from the TAT register, in terms of serial clock cycles) as shown in Figure 20.9 and Figure 20.10.

**20.3.3.2.2.1 RE to DE Turnaround Time** UART inserts the wait state (as programmed in TAT[31:16] times serial clock) before switching to transmit mode from receive mode as shown in the Figure 20.9 (applicable only when TCR[4:3] =1 or 2 (XFER\_MODE).)

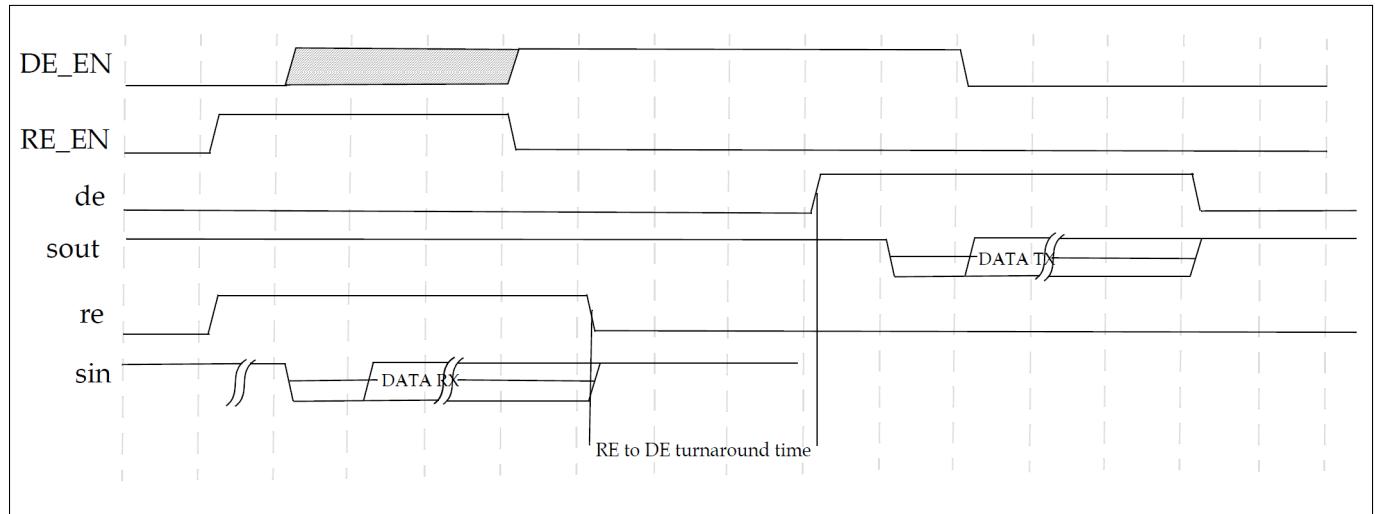


Figure 20.9: RE to DE Turnaround Time

**20.3.3.2.3 DE to RE Turnaround Time** UART inserts the wait state (as programmed in TAT[15:0] times serial clock) before switching to receive mode from transmit mode as shown in the Figure 20.10 (applicable only when TCR[4:3] =1 or 2 (XFER\_MODE).)

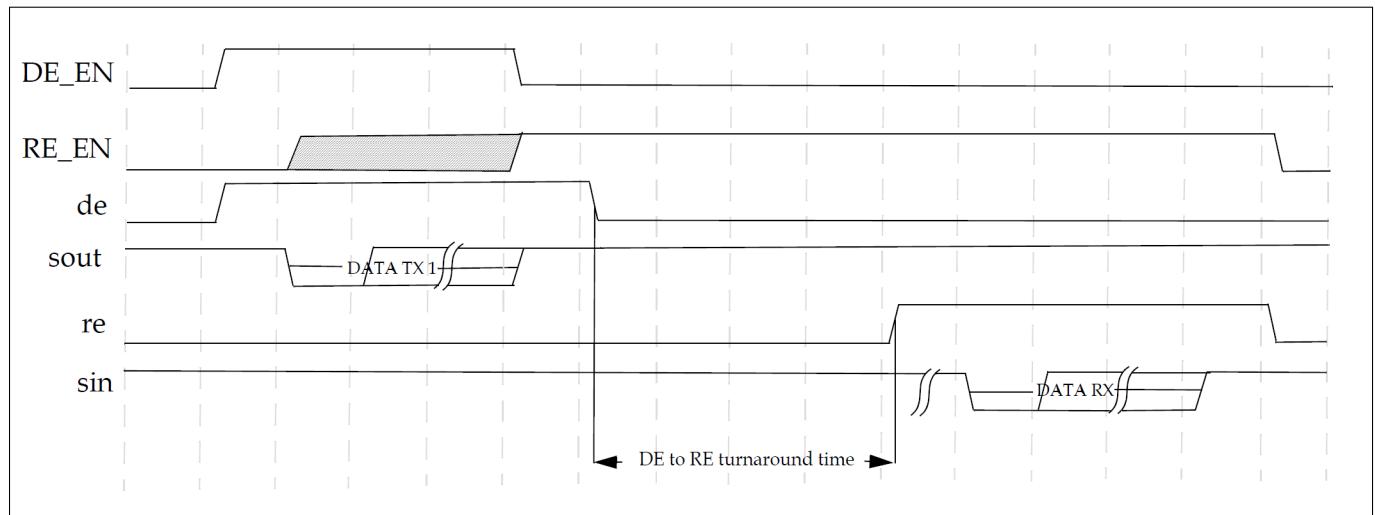


Figure 20.10: DE to RE Turnaround Time

**20.3.3.2.4 Hardware-Controlled Half Duplex Mode** The hardware-controlled half duplex mode supports either transmit or receive transfers at a time but not both simultaneously. If both 'DE Enable' and 'RE Enable' bits of Driver output enable (DE\_EN) and Receiver output enable (RE\_EN) registers are enabled, the switching between transmit to receive or receive to transmit is automatically done by the hardware based on the empty condition of Tx-FIFO.

In hardware-controlled half duplex mode, the de signal:

- Goes active if the following conditions are satisfied:
  - The DE Enable (DE\_EN[0]) field of Driver Output Enable Register is set to 1.

- Transmitter Holding Register is not empty in non-FIFO mode or transmitter FIFO is not empty in FIFO mode.
- If any receive transfer is ongoing, then the signal waits until receive is finished and after the turnaround time counter ('re to de') has elapsed.
- Goes inactive if the following conditions are satisfied
  - The current ongoing transmitting serial transfer is completed.
  - Either transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in nonFIFO mode or the DE Enable (DE\_EN[0]) of Driver output Enable Register is set to 0.

In hardware-controlled half duplex mode, the re signal:

- Goes active if the following conditions are satisfied:
  - When RE Enable (RE\_EN[0]) field of Receiver Output Enable Register is set to 1.
  - Either transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in nonFIFO mode.
  - If any transmit transfer is ongoing, then the signal waits until transmit is finished and after the turnaround time counter ('de to re') has elapsed.
- Goes inactive under the following conditions:
  - The current ongoing receive serial transfer has completed.
  - Either transmitter FIFO is non-empty in FIFO mode or Transmitter Holding Register is non empty in non-FIFO mode or the RE Enable (RE\_EN[0]) of Receiver output Enable Register is set to 0.

In this mode, the hardware ensures that a proper turnaround time is maintained while switching from 're' to 'de' or from 'de' to 're' (value of turnaround is obtained from the TAT register, in terms of serial clock cycles) as shown in Figure 20.9 and Figure 20.10.

### 20.3.3.3 Sample Scenarios

Consider a scenario in which the UART controller is receiving 3 characters and another UART device is sending those characters. While the 1st character is being received by the UART controller, if the software writes into the TX FIFO of the UART controller, then at the end of the first character UART controller will switch the mode from receive to transmit. UART will de-assert 're' and assert 'de' signal. This will make the UART controller not to receive the subsequent characters.

Hence, in hardware switching half duplex mode, the user has to ensure that complete receive data has been received before writing in to the Tx-FIFO to avoid missing of receive characters.

Following sections explain the behavior of UART in XFER\_MODE=2 for different scenarios.

#### 20.3.3.3.1 Normal Scenario of Transmission

Figure 20.11 is a sample scenario for normal transmission.

Figure 20.11 shows the following activities at various points in this scenario:

1. At this point, reset is removed, and de and re signals are driven to their configured reset values (UART\_DE\_POL/UART\_RE\_POL).
2. At this point, the software programs DE\_EN and RE\_EN register to 1. At this point in time, tx\_fifo\_empty \* is 1 indicating that there is no data in TX FIFO. Hence, the 'de' signal remains deasserted and 're' gets asserted. \* tx\_fifo\_empty is internal signal of UART

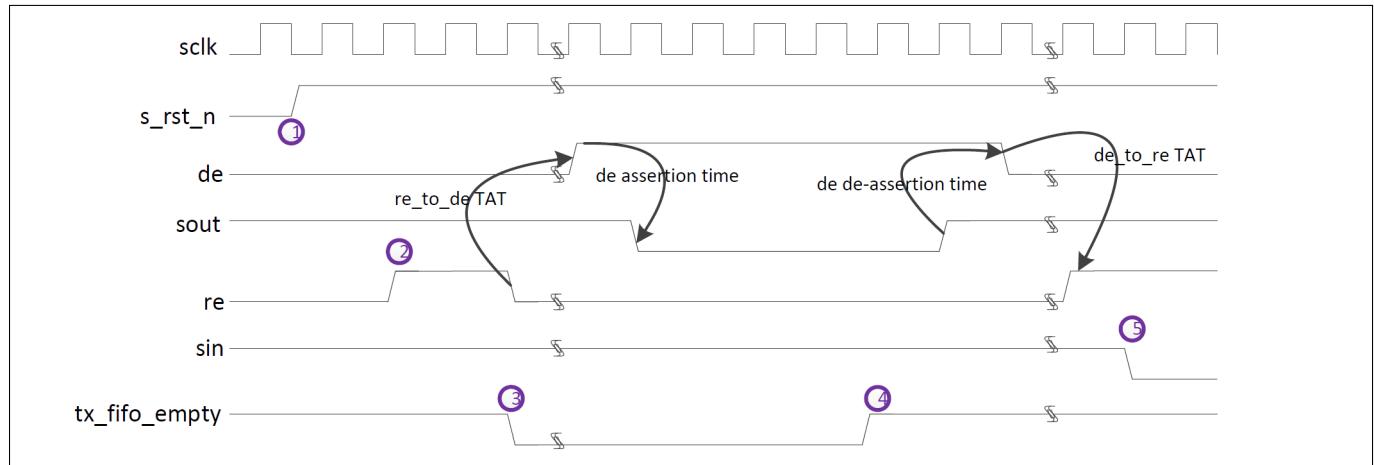


Figure 20.11: Scenario When XFER\_MODE=2

- At this point, the software fills the TX FIFO and there is no ongoing Receive transfer. Therefore, the 're' signal goes low. However, the UART controller waits until 're\_to\_de' TAT value before asserting 'de' signal. After the 'de' gets asserted, the transmission of character starts considering the 'de-asserting timing'.
- At this point, TX FIFO becomes empty. After transmitting the current character, UART will de-assert the 'de' signal (after de assertion time). UART controller waits until 'de\_to\_re' TAT values before asserting 're' signal back.
- At this point, UART controller starts receiving the character.

**20.3.3.3.2 Scenario When Receive is in Progress While TX FIFO is Being Filled** In this scenario, TX FIFO is filled when a character is being received. In this case, UART is expected to wait till the current character is finished before changing the role and start transmitting.

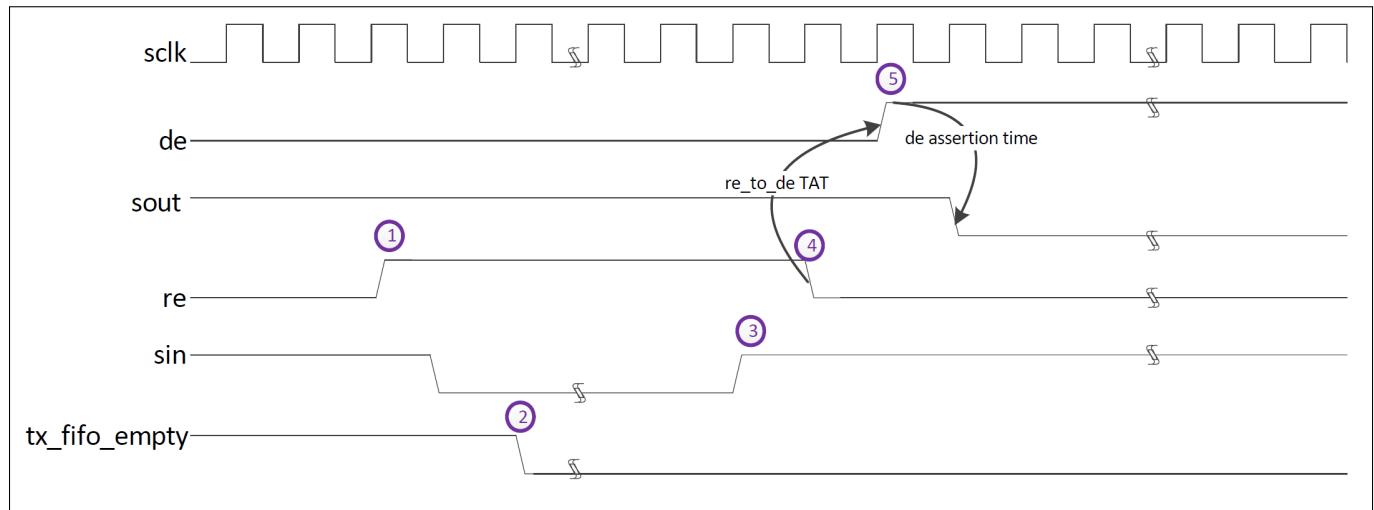


Figure 20.12: Receive in Progress, When TX FIFO is Filled

Figure 20.12 shows the following activities at various points in this scenario:

- The software programs DE\_EN and RE\_EN to 1, thereby asserting the 're' signal. After this, the UART controller

starts receiving the character.

2. The software programs TX FIFO thereby making 'tx\_fifo\_empty' to go low. However, the UART controller waits until the current character is received before asserting the 'de' signal.
3. The incoming character is fully received.
4. The 're' signal gets de-asserted, after the STOP bit is fully received.
5. After the 're\_to\_de' TAT, the 'de' signal gets asserted and the UART controller starts transmitting after DET timings.

**20.3.3.3.3 TX FIFO Filled Before Enabling DE\_EN and RE\_EN Registers** In this case, TX FIFO is filled prior to enabling DE\_EN or RE\_EN. The UART controller enables the 'de' instead of 're' in this case because TX FIFO already has the data to transmit.

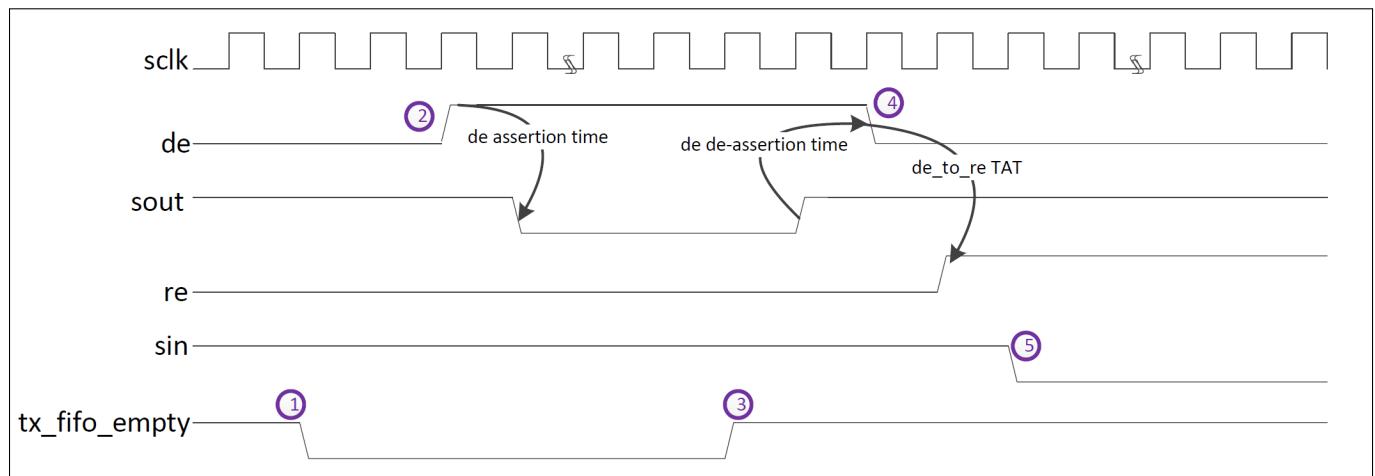


Figure 20.13: TX FIFO is Filled Before Enabling DE/RE

Figure 20.13 shows the following activities at various points in this scenario:

1. The software programs the TX FIFO thereby making 'tx\_fifo\_empty' signal to go low.
2. The software programs 'DE\_EN' and 'RE\_EN' to 1. As the data is already present in the TX FIFO, UART controller asserts the 'de' signal. UART stars sending the character after the DET timings.
3. TX FIFO becomes empty.
4. The 'de' signal gets de-asserted after the DET timing. After 'de\_to\_re' TAT, 're' signal gets asserted.
5. UART controller starts receiving the incoming character.

### 20.3.4 Fractional Baud Rate Support

UART supports fractional baud rate that enables a user to program the fractional part of the divisor value to generate fractional baud rate that results in reduced frequency error. The UART interface usage has been evolving to include ever increasing baud rate speeds. The UART needs to be software configurable to handle the baud rates within 2% frequency error.

The Baud rate of UART is controlled by sclk in asynchronous serial clock (CLOCK\_MODE=2) implementation or pclk in single clock implementation (CLOCK\_MODE=1) and the Divisor Latch Register (DLH and DLL).

The baud rate is determined by the following factors:

- Serial clock operating frequency (sclk in Asynchronous serial clock implementation or pclk in single clock implementation)
- The desired baud rate.
- The baud rate generator divisor value, DIVISOR (composed of DLH & DLL registers).
- The acceptable Baud-rate error, %ERROR

The equation to calculate the baud rate is as follows:

$$\text{BaudRate} = \frac{\text{SerialClockOperatingFrequency}}{(16 \times \text{DIVISOR})} \quad (20.1)$$

Where,

DIVISOR – Number (in hexadecimal) to program the DLL and DLH.

Serial clock frequency – Frequency at sclk or pclk pin of UART.

From Equation 20.1, DIVISOR can be calculated as:

$$\text{DIVISOR} = \frac{\text{SerialClockOperatingFrequency}}{(16 \times \text{BaudRate})} \quad (20.2)$$

Also from Equation 20.1, it can also be shown that:

$$\text{Serialclockfrequency} = \text{BaudRate} \times 16 \times \text{DIVISOR} \quad (20.3)$$

The Error between the Baud rate and Baud rate (selected) is given as:

$$\text{PercentageERROR} = \frac{|\text{BaudRate} - \text{BaudRate(selected)}|}{\text{BaudRate}} \times 100 \quad (20.4)$$

Note

Fractional Baud rate is supported only when the source license DWC-APB-Advanced-Source exists.

Configuration of the UART for Fractional Baud Rate does the following:

- The configurable parameter DLF\_SIZE is used to choose the width of the register that stores fractional part of the divisor.

- The fractional value of the divisor is programmed in the Divisor Latch Fraction Register (DLF) register. The fractional value is computed by using the (Divisor Fraction value)/(2^DLF\_SIZE) formula. Table 20.1 shows fractional values when the DLF\_SIZE=4.

DLF Value	Fraction	Fractional Value
0000	0/16	0.0000
0001	1/16	0.0625
0010	2/16	0.125
0011	3/16	0.1875
0100	4/16	0.25
0101	5/16	0.3125
0110	6/16	0.375
0111	7/16	0.4375
1000	8/16	0.5
1001	9/16	0.5625
1010	10/16	0.625
1011	11/16	0.6875
1100	12/16	0.75
1101	13/16	0.8125
1110	14/16	0.875
1111	15/16	0.9375

**Table 20.1:** Divisor Latch Fractional Values

The programmable fractional baud rate divisor enables a finer resolution of baud clock than the conventional integer divider. The programmable fractional baud clock divider allows for the programmability of both an integer divisor as well as fractional component. The average frequency of the baud clock from the fractional baud rate divisor is dependent upon both the integer divisor and the fractional component, thereby providing a finer resolution to the average frequency of the baud clock.

$$\text{BaudRateDivisor} = \frac{\text{SerialClockFrequency}}{(16 \times \text{RequiredBaudRate})} = BRD_I + BRD_F \quad (20.5)$$

Where,

$BRD_I$  - Integer part of the divisor.

$BRD_F$  - Fractional part of the divisor.

#### 20.3.4.1 Fractional Division Used to Generate Baud Clock

Fractional division of clock is used by the N/N+1 divider, where N is the integer part of the divisor. N/N+1 division works on the basis of achieving the required average timing over a long period by alternating the division between two numbers. If N=1 and ratio of N/N+1 is same, which means equal number of divide by 1 and divide by 2 over a

period of time, average time period would come out to be divided by 1.5. Varying the ratio of N/N+1 any value can be achieved above 1 and below 2.

#### 20.3.4.2 Calculating the Fractional Value Error

Following is a sample for calculating the fractional value error.

Consider the following values:

- Required Baud Rate (RBR) = 4454400
- Serial Clock (SCLK) = 133MHz
- DLF\_SIZE = 4

Then, as per equation 20.5, Baud Rate Divisor (BRD) is as follows:

$$BRD = \frac{133}{16 \times 4454400} = 1.866132364 \quad (20.6)$$

In equation 20.6, the integer and fractional parts are as follows:

- Integer part ( $BRD_I$ ) = 1
- Fractional part ( $BRD_F$ ) = 0.866132364

Therefore, Baud Rate Divisor Latch Fractional Value (DLF) is as follows:

$$DLF = BRD_F \times 2^{(DLF\_SIZE)} = 0.86613236416 = 13.858117824 = 14(\text{roundoff value}) \quad (20.7)$$

The Generated Baud Rate Divider (GD) is as follows:

$$GD = BRD_I + \frac{DLF}{2^{DLF\_SIZE}} = 1 + \frac{14}{16} = 1.875 \quad (20.8)$$

Therefore, the Generated Baud Rate (GBR) is as follows:

$$GBR = \frac{\text{SerialClock}}{(16 \times GD)} = \frac{133}{(16 \times 1.875)} = 4433333.333 \quad (20.9)$$

Now the error is calculated as follows:

$$Error = \frac{GBR - RBR}{RBR} = 0.004729 \quad (20.10)$$

The error percentage is as follows:

$$Error\% = 0.004729 \times 100 = 0.473 \quad (20.11)$$

**20.3.4.2.1 Timing Waveforms** If serial clock is 25 MHz and Baud rate required = 892857, divisor comes out to be 1.75. Without a fractional division a value of 1 or 2 will result in baud rate of 1562500 or 781250 which is more than 2% frequency error. However, if we divide 12 clocks by 2 and then 4 clocks by 1, over an average period of 16 clocks we'll achieve division by 1.75. Using this divisor baud rate of 892857 can be achieved.

As shown in the Figure 20.14, the fractional baud clock is generated between N(1) and N+1(2) values to generate the fractional baud rate of 1.75 to achieve the divisor baud rate of 892857 with 0% frequency error compared to 12.49% frequency error in integer baud clock generator.

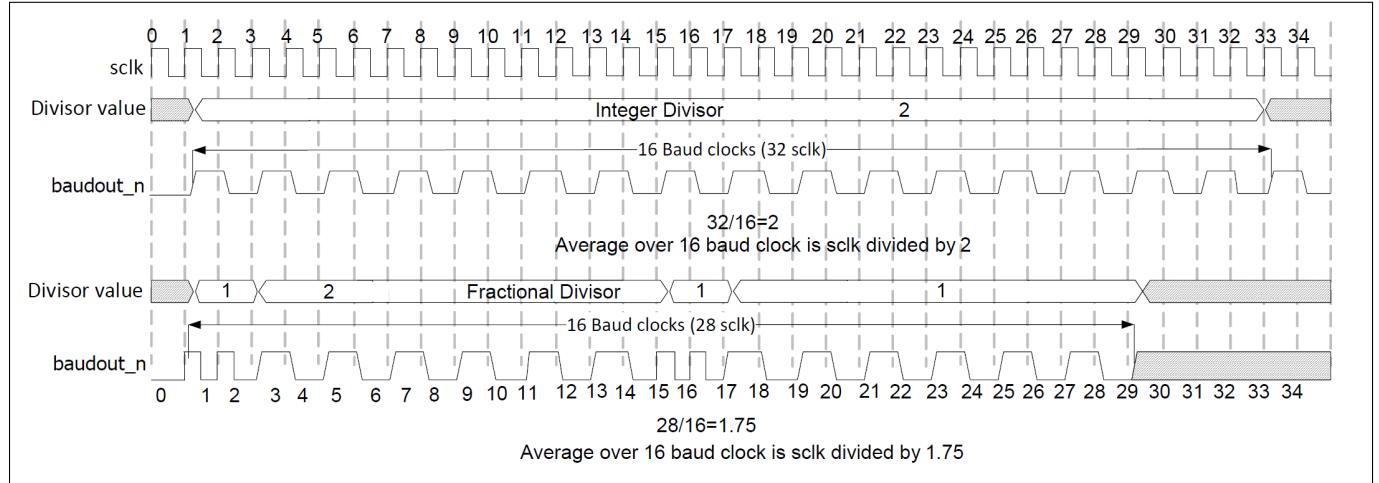


Figure 20.14: Example of Integer and Fractional Division Over 16 Clock Periods

### 20.3.5 IrDA 1.0 SIR Protocol

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. IrDA 1.0 SIR mode specifies a maximum baud rate of 115.2 Kbaud.

#### Attention

Information provided on IrDA SIR mode in this section assumes that the reader is fully familiar with the IrDa Serial Infrared Physical Layer Specifications. This specification can be obtained from the following website:

<http://www.irda.org>

The data format is similar to the standard serial—sout and sin—data format. Each data character is sent serially in this order:

1. Begins with a start bit
2. Followed by 8 data bits
3. Ends with at least one stop bit

Thus, the number of data bits that can be sent is fixed. No parity information can be supplied, and only one stop bit is used in this mode. Trying to adjust the number of data bits sent or enable parity with the Line Control Register (LCR) has no effect.

Configuration of the UART for IrDA 1.0 SIR does the following:

- Bit 6 of the Mode Control Register (MCR) enables or disables the IrDA 1.0 SIR mode.
- Disabling IrDA SIR mode causes the logic to not be implemented; the mode cannot be activated, which reduces total gate counts.
- When IrDA SIR mode is enabled and active, serial data is transmitted and received on the sir\_out\_n and sir\_in ports, respectively.

#### Note

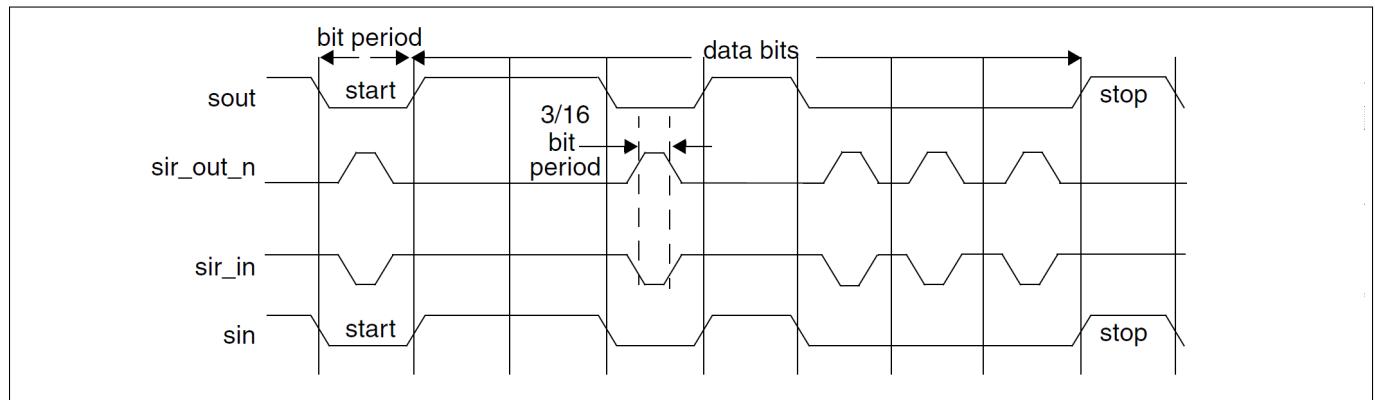
To enable SIR mode, write the appropriate value to the MCR register before writing to the LCR register. For details of the recommended programming sequence, refer to “Programing Examples”.

Transmission or non-transmission of a single infrared pulse indicates the following:

- Transmitting a single infrared pulse indicates logic 0
- Non-transmission of a pulse indicates logic 1

The width of each pulse is 3/16ths of a normal serial bit time. Thus, each new character begins with an infrared pulse for the start bit. However, received data is inverted from transmitted data due to infrared pulses energizing the photo transistor base of the IrDA receiver, which pulls its output low. This inverted transistor output is then fed to the UART sir\_in port, which gives it the correct UART polarity.

Figure 20.15 shows the timing diagram for the IrDA SIR data format in comparison to standard serial format.



**Figure 20.15:** IrDA SIR Data Format

As previously mentioned, the UART can be configured to support a low-power reception mode. When the UART is configured in this mode, it is possible to receive SIR pulses of 1.41 microseconds (minimum pulse duration), as well as nominal 3/16 of a normal serial bit time. In order to use this lowpower reception mode, you must program the Low Power Divisor Latch (LPDLL/LPDLH) registers.

For all sclk frequencies greater than or equal to 7.37MHz, pulses of 1.41uS are detectable; these pulses comply with the requirements of the Low Power Divisor Latch registers. However, there are several values of sclk that do not allow detection of such a narrow pulse, as indicated in Table 20.2.

SCLK	Low Power Divisor Register Value	Min Pulse Width for Detection *
1.84MHz	1	3.77uS
3.69MHz	2	2.086uS
5.53MHz	3	1.584uS

\* 10% has been added to the internal pulse width signal to cushion the effect of pulse reduction due to the synchronization and data integrity logic so that a pulse slightly narrower than these may be detectable.

**Table 20.2:** Narrow Pulse Exceptions

When IrDA SIR mode is enabled, the UART operates in a manner similar to when the mode is disabled, with one exception: data transfers can only occur in half-duplex fashion when IrDA SIR mode is enabled. This is because the IrDA SIR physical layer specifies a minimum of 10ms delay between transmission and reception; this 10ms delay must be generated by software.

### 20.3.6 FIFO Support

You can configure the UART to implement FIFOs that buffer transmit and receive data; this is illustrated in Figure 20.1. If FIFO support is not selected, then no FIFOs are implemented and only a single receive data byte and transmit data byte can be stored at a time in the RBR and THR registers; this implies a 16450-compatible mode of operation. However, in this mode of operation, most of the enhanced features are unavailable.

In FIFO mode, the FIFOs can be selected to be either of the following:

- External customer-supplied FIFO RAMs
- Internal DesignWare D-flip-flop-based RAMs (DW\_ram\_r\_w\_s\_dff)

If the configured FIFO depth is greater than 256, the FIFO memory selection is restricted to be external. Additionally, selecting internal memory restricts the Memory Read Port Type to D-flip-flop-based Synchronous read port RAMs.

When external RAM support is chosen, either synchronous or asynchronous RAMs can be used. Asynchronous RAM provides read data during the clock cycle that has the memory address and read signals active, for sampling on the next rising clock edge. Synchronous single stage RAM registers the data at the current address out and is not available until the next clock cycle; that is, the second rising clock edge.

Figure 20.16 shows the timing diagram for both asynchronous and synchronous RAMs.

#### Note

This timing diagram illustrated in Figure 20.16 assumes the RAM has a chip select port that is tied to an active value; therefore, the chip is always enabled. This is why the second synchronous read data appears at the same cycle as the asynchronous read data; that is, the address for the second read has been sampled along with the chip select on an earlier edge. Once the tx\_ram\_re\_n output enable asserts the data, the value on the register output is seen on that same cycle.

Similarly, you can use synchronous RAM for writes, which registers the data at the current address out.

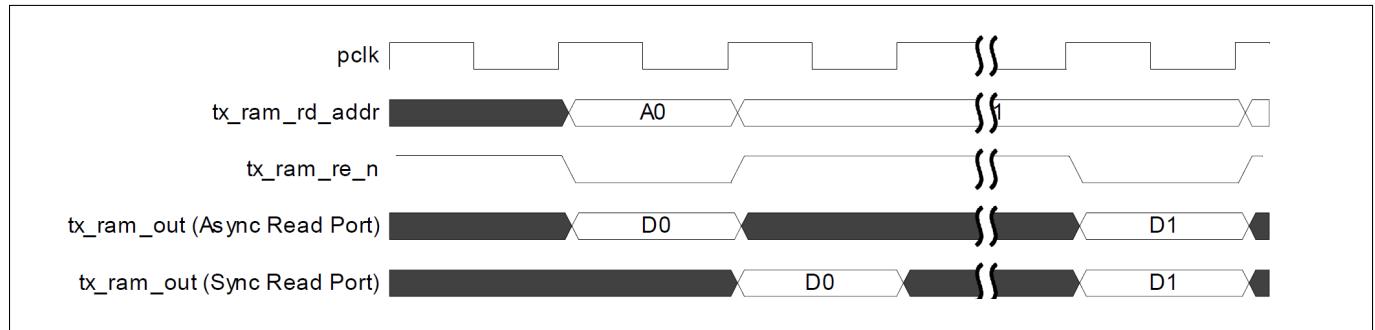
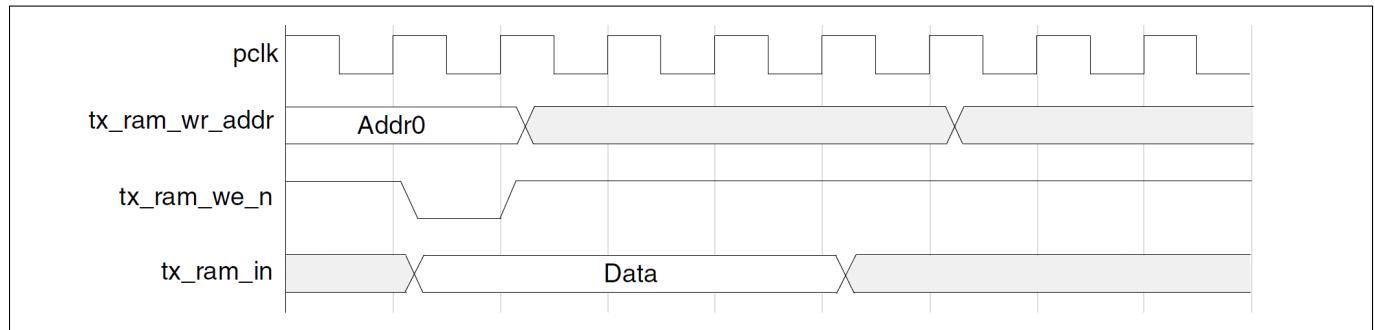
**Figure 20.16:** Timing for RAM Reads

Figure 20.17 shows the timing diagram for RAM writes.

**Figure 20.17:** Timing for RAM Writes

When FIFO support is selected, an optional programmable FIFO Access mode is available for test purposes, which allows:

- Receive FIFO to be written by master
- Transmit FIFO to be read by master

When FIFO Access mode is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts.

When FIFO Access mode has been selected it can be enabled with the FIFO Access Register (FAR[0]). Once enabled, the control portions of the transmit and receive FIFOs are reset and the FIFOs are treated as empty.

Data can be written to the transmit FIFO as normal; however no serial transmission occurs in this mode—normal operation halted—and thus no data leave the FIFO. The data that has been written to the transmit FIFO can be read back with the Transmit FIFO Read (TFR) register, which when read gives the current data at the top of the transmit FIFO.

Similarly, data can be read from the receive FIFO as normal. Since the normal operation of the USART is halted in this mode, data must be written to the receive FIFO so the data can be read back.

Data is written to the receive FIFO using the Receive FIFO Write (RFW) register. The upper two bits of the 10-bit register are used to write framing error and parity error detection information to the receive FIFO, as follows:

- RFW[9] indicates framing error

- RFW[8] indicates parity error

Although these bits cannot be read back through the Receive Buffer Register, they can be checked by reading the Line Status Register and checking the corresponding bits when the data in question is at the top of the receive FIFO.

### 20.3.7 Clock Support

The UART can be configured to have either one system clock (pclk) or two system clocks (pclk and sclk). The second asynchronous serial clock (sclk) accommodates accurate serial baud rate settings, as well as APB bus interface requirements. When using a single-system clock, available system clock settings for accurate baud rates are greatly restricted.

When a two-clock design is chosen, a synchronization module is implemented for synchronization of all control and data across the two-system clock boundaries; this is illustrated in Figure ??.

The RTL diagram for the data synchronization module is shown in Figure 20.18; this module can have pending data capability.

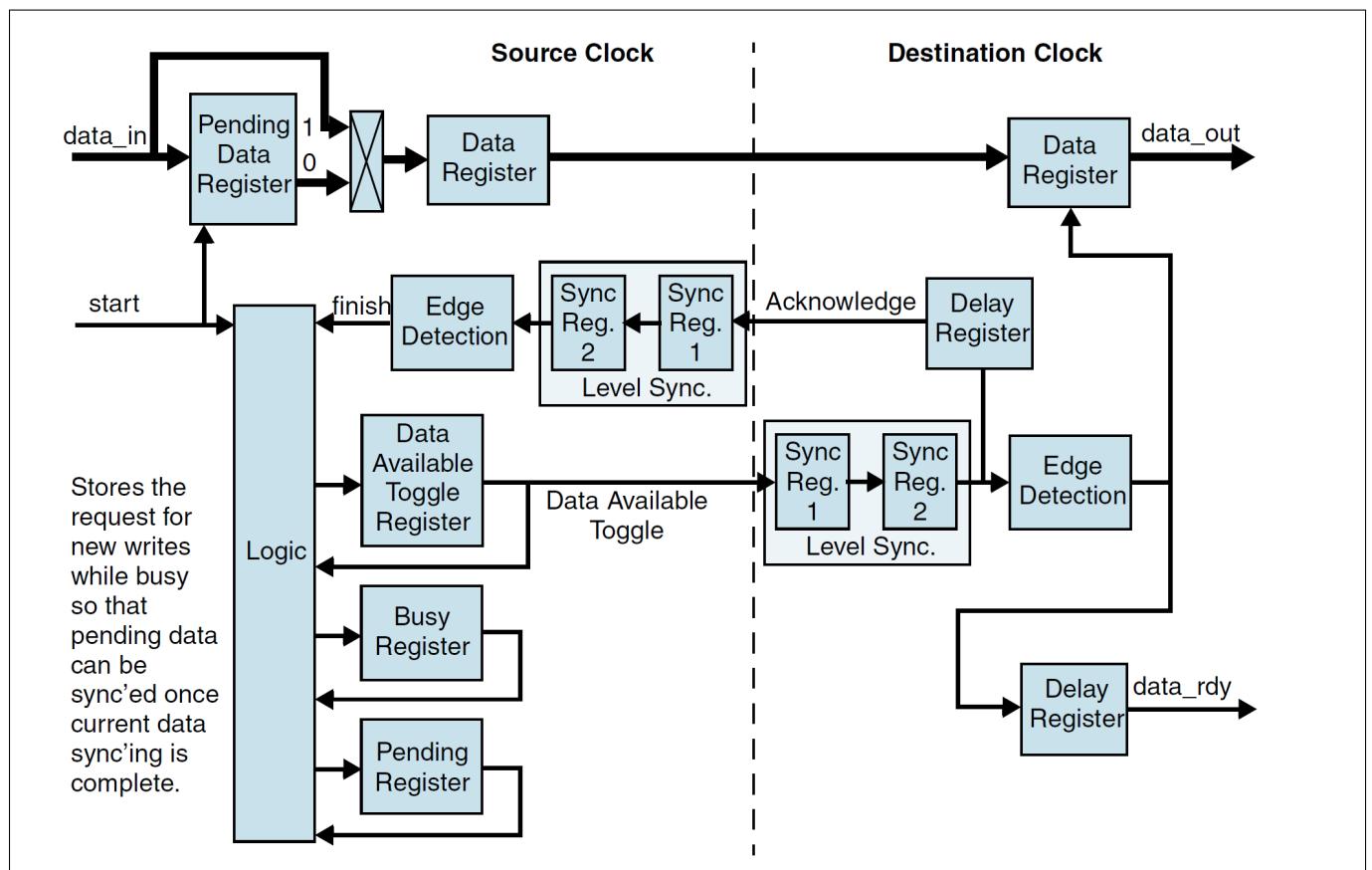
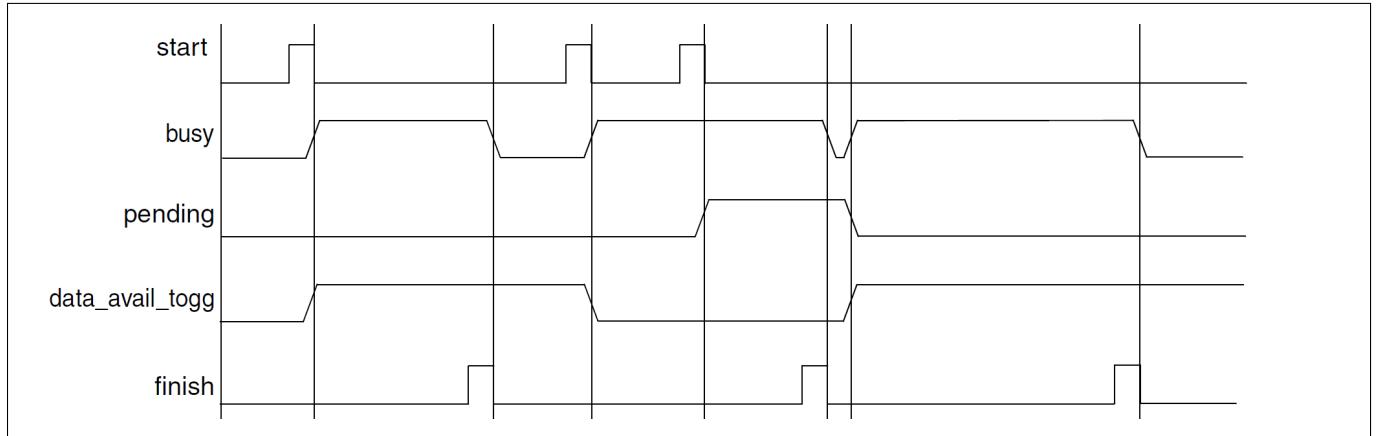


Figure 20.18: RTL Diagram of Data Synchronization Module

The timing diagram shown in Figure 20.19 shows the data synchronization process.

The arrival of new source domain data is indicated by the assertion of start. Since data is now available for



**Figure 20.19:** Timing Diagram for Data Synchronization Module

synchronization, the process is started and busy status is set. If start is asserted while busy and pending data capability has been selected, the new data is stored.

When no longer busy, the synchronization process starts on the stored pending data. Otherwise the busy status is removed when the current data has been synchronized to the destination domain and the process continues. If only one clock is implemented, all synchronization logic is absent and signals are simply passed through this module.

There are two types of signal synchronization:

- Data-synchronized signals – full synchronization handshake takes place on signals
- Level-synchronized signals – signals are passed through two destination clock registers

Both synchronization types incur additional data path latencies. However, this additional latency has no negative affect on received or transmitted data, other than to limit how much faster sclk can be in relation to pclk for back-to-back serial communications with no idle assertion.

A serial clock that exceeds this limit does not leave enough time for a complete incoming character to be received and pushed into the receiver FIFO. To ensure that you do not exceed the limit, the following equation must hold true:

$$((2 \times \text{pclk\_cycles}) + 4) < (39 \times (\text{BaudDivisor})) \quad (20.12)$$

Where:

pclk\_cycles is expressed in sclk cycles

For example, if the Baud Divisor is programmed to 1 and a serial clock is 18 times faster than the pclk signal, the equation becomes:

$$((2 \times 18) + 4) < (39 \times 1) \geq 40 < 39 \quad (20.13)$$

Thus the equation does not hold true, and the ratio 18:1 (sclk:pclk) exceeds the limit at this Baud rate.

Here are a few things to keep in mind:

- A divisor greater than 1 at a clock ratio of 18:1 (sclk:pclk) does not cause data corruption issues due to synchronization, as the synchronization process has more time to transfer the received data to the peripheral clock domain before the next character bit is received. In most cases, however, the pclk signal is faster than sclk, so this should never be an issue.
- There is slightly more time required after initial serial control register programming before serial data can be transmitted or received.
- The serial clock modules must have time to see new register values and reset their respective state machines. This total time is guaranteed to be no more than eight clock cycles of the slower of the two system clocks. Therefore, no data should be transmitted or received before this maximum time expires, after initial configuration.

Each NOP usually takes one bus cycle to retire. However, the actual number of NOPs that need to be inserted in the assembly code is dependent on the maximum number of instructions that can be retired in a single cycle. So for example, if the processor uses a 4-dispatch pipe, then four NOPs could potentially retire in one bus cycle. Assuming that the next opcode (NOP) is fetched as per the slower clock—with eight clock cycles of the slower clock as the reference—a minimum of thirty-two NOPs need to be included in the assembly code after a software reset.

In systems where only one clock is implemented, there are no additional latencies.

### 20.3.8 Back-to-Back Character Stream Transmission

This section describes:

- Scenarios under which the UART is capable of transmitting back-to-back characters on the serial interface, with no idle time between them
- Worst-case idle time that exists between back-to-back characters

When the Transmit FIFO contains multiple data entries, the UART transmits the characters in the FIFO back-to-back on the serial bus. However, if the CLOCK\_MODE configuration parameter equals 2, synchronization delays in the UART can cause an IDLE period between the end of the current STOP bit and the beginning of the next START bit; this appears as an extended STOP bit duration on the serial bus.

#### 20.3.8.1 Dual Clock Mode

When the CLOCK\_MODE parameter equals 2—indicating an asynchronous relationship between pclk and sclk—the UART has a synchronization delay between the transmitter in the sclk domain and the TX FIFO in the pclk domain when querying if another character is ready for transmission. The transmitter begins the handshake one baud clock cycle before the end of the current STOP bit. The duration of the synchronization delay is given by the following equations:

$$\text{sync\_delay} = (1\text{sclk} + 3\text{pclk}) + 1\text{pclk} + (1\text{pclk} + 3\text{sclk})$$

$$\text{sync\_delay} = 4\text{sclk} + 5\text{pclk}$$

If the sync\_delay duration is longer than one baud clock period, an IDLE period will be inserted between the end of a STOP bit and the beginning of the next START bit.

To prevent insertion of the IDLE period, the following condition must be true:

$$\text{sync\_delay} \leq \text{bclk\_period}$$

The baud clock period is given by the following equation:

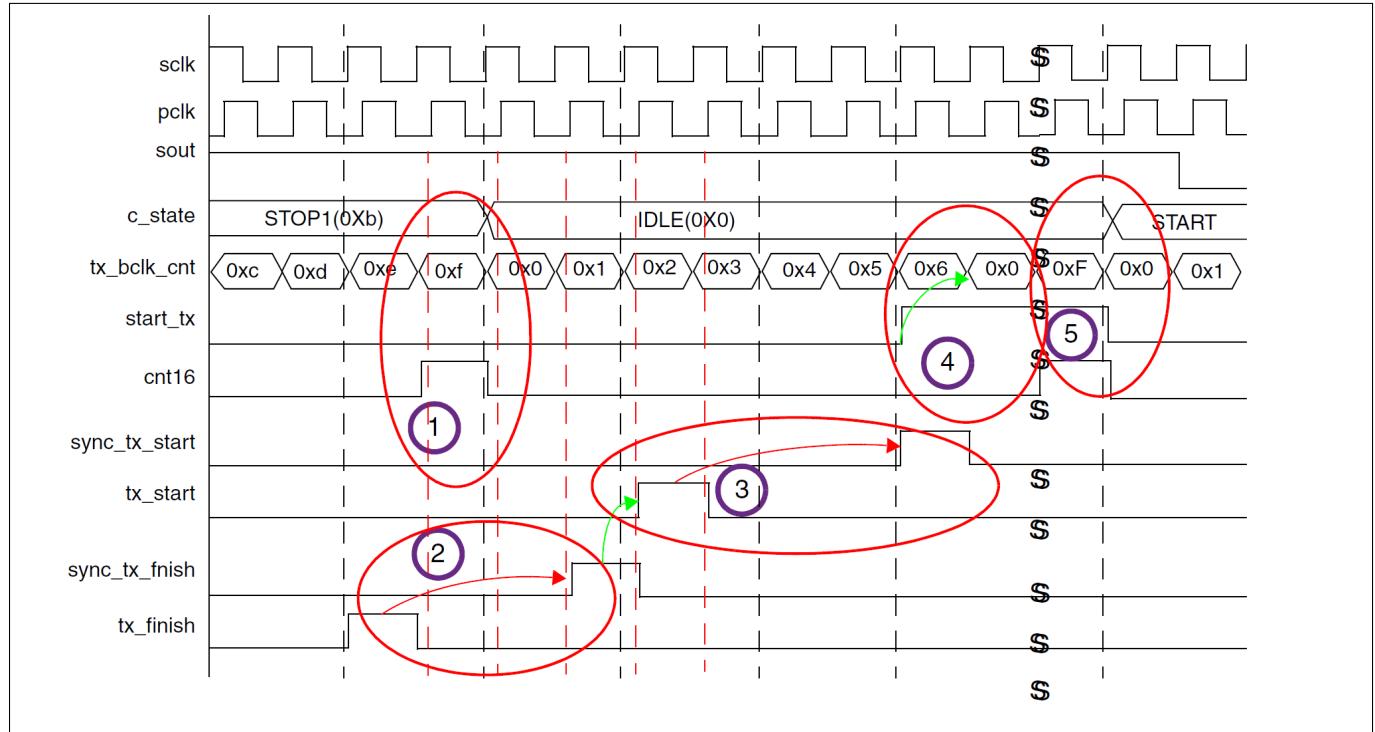
$$\text{bclk\_period} = \{\text{DLH}, \text{DLL}\} * \text{sclk}$$

The worst case timing of the inserted IDLE period is given by:

$$\text{worst\_case\_idle\_duration} = \text{sync\_delay} + (15 * \text{bclk\_period})$$

The `worst_case_idle_duration` can be added to the programmed STOP bit duration to give the overall STOP bit period.

Figure 20.20 illustrates an example of character finish to character start delay.



**Figure 20.20:** Character Finish to Character Start Delay

1. The baud divisor is set to 1 ( $\{\text{DLH}, \text{DLL}\} = 1$ ), so every `sclk` is a baud clock cycle. The transmit state machine changes state every sixteen baud clocks—eight in the case of a half STOP bit. At this point in Figure 20.20, after 16 baud clock cycles of the `STOP1` state, the state machine enters the `IDLE` state on the next cycle because `start_tx` is not yet asserted.
2. One baud clock before the end of the `STOP` state, the transmit state machine decodes that the current character is complete and asserts `tx_finish`, which is synchronized to the `pclk` domain to become `sync_tx_finish`; this synchronization accounts for the “ $1\text{sclk} + 3\text{pclk}$ ” term in `sync_delay`.

3. In the pclk domain, there is a one-pclk cycle delay—"1pclk" term in sync\_delay—before the signal tx\_start is asserted from the assertion of sync\_tx\_finish. Tx\_start must then be synchronized to the sclk domain—"1pclk + 3sclk" term in sync\_delay—to instruct the state machine to commence the START bit of the next character.
4. Start\_tx asserts in the sclk domain, and causes the baud clock counter (tx\_bclk\_cnt) to go to 0.
5. Once sixteen baud clocks have been counted, the state machine can transition into the START state, and one cycle later sout is de-asserted.

#### 20.3.8.2 Single Clock Mode

If CLOCK\_MODE equals 1, there is no idle time between back-to-back characters if data is ready in the transmit FIFO. In this case, because sync\_delay equals one pclk as described in "Dual Clock Mode", the requirement to avoid idle time between consecutive characters is met for all {DLH,DLL} values.

$$\text{sync\_delay} \leq \{\text{DLH},\text{DLL}\}^*\text{sclk}$$

For example, when {DLH, DLL} equals 1 (bearing in mind that when CLOCK\_MODE = 1 : pclk = sclk), then 1 pclk  $\leq 1^*\text{pclk}$

#### 20.3.9 Interrupts

Assertion of the UART interrupt output signal (intr)—a positive-level interrupt—occurs whenever one of the several prioritized interrupt types are enabled and active.

When an interrupt occurs, the master accesses the IIR register.

The following interrupt types can be enabled with the IER register:

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)
- Modem Status
- Busy Detect Indication

These interrupt types are explained in detail in Table 20.3.

Interrupt Set and Reset Functions							
Bit3	Bit2	Bit1	Bit0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	-	None	None	-

**Table 20.3: Narrow Pulse Exceptions (Continued on next page)**

continued from previous page							
Interrupt ID		Interrupt Set and Reset Functions					
Bit3	Bit2	Bit1	Bit0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	0	Highest	Receiver line status	Overrun/parity/ framing errors, break interrupt, or address received interrupt	<p>For Overrun/parity/framing/break interrupt reset control, the behavior is as follows:</p> <ul style="list-style-type: none"> <li>(1) If LSR_STATUS_CLEAR=0 (RBR Read or LSR Read), then the status is cleared on:           <ul style="list-style-type: none"> <li>- Reading the line status register.</li> </ul> </li> <li>Or</li> <li>- In addition to an LSR read, the Receiver line status is also cleared when RX_FIFO is read.</li> </ul> <ul style="list-style-type: none"> <li>(2) If LSR_STATUS_CLEAR=1 (LSR Read), the status is cleared only on:           <ul style="list-style-type: none"> <li>- Reading the line status register.</li> </ul> </li> <li>(3) For address received interrupt, the status is cleared on:           <ul style="list-style-type: none"> <li>- Reading the line status register.</li> </ul> </li> </ul>
0	1	0	0	Second	Received data available	Receiver data available (non-FIFO mode or FIFOs disabled) or RCVR FIFO trigger level reached (FIFO mode and FIFOs enabled)	Reading the receiver buffer register (non-FIFO mode or FIFOs disabled) or the FIFO drops below the trigger level (FIFO mode and FIFOs enabled)
1	1	0	0	Second	Character timeout indication	No characters in or out of the RCVR FIFO during the last 4 character times and there is at least 1 character in it during this time	Reading the receiver buffer register

Table 20.3: Narrow Pulse Exceptions (Continued on next page)

continued from previous page							
Inter-ID	Interrupt Set and Reset Functions						
Bit3	Bit2	Bit1	Bit0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	1	0	Third	Transmit holding register empty	Transmitter holding register empty (Prog. THRE Mode disabled) or XMIT FIFO at or below threshold (Prog. THRE Mode enabled)	Reading the IIR register (if source of interrupt); or, writing into THR (FIFOs or THRE Mode not selected or disabled) or XMIT FIFO above threshold (FIFOs and THRE Mode selected and enabled).
0	0	0	0	Fourth	Modem status	Clear to send or data set ready or ring indicator or data carrier detect. Note that if auto flow control mode is enabled, a change in CTS (that is, DCTS set) does not cause an interrupt.	Reading the Modem status register
0	1	1	1	Fifth	Busy detect indication	UART_16550_COMPATIBLE = NO and master has tried to write to the Line Control Register while the DW_apb_uart is busy (USR[0] is set to 1).	Reading the UART status register

**Table 20.3:** Narrow Pulse Exceptions

### 20.3.10 Auto Flow Control

The UART can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available; if FIFOs are not implemented, this mode cannot be selected. When Auto Flow Control is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When Auto Flow Control mode is selected, it can be enabled with the Modem Control Register (MCR[5]).

Figure 20.21 shows a block diagram of the Auto Flow Control functionality.

Auto RTS and Auto CTS are described as follows:

- Auto RTS – Becomes active when the following occurs:
  - Auto Flow Control is selected during configuration

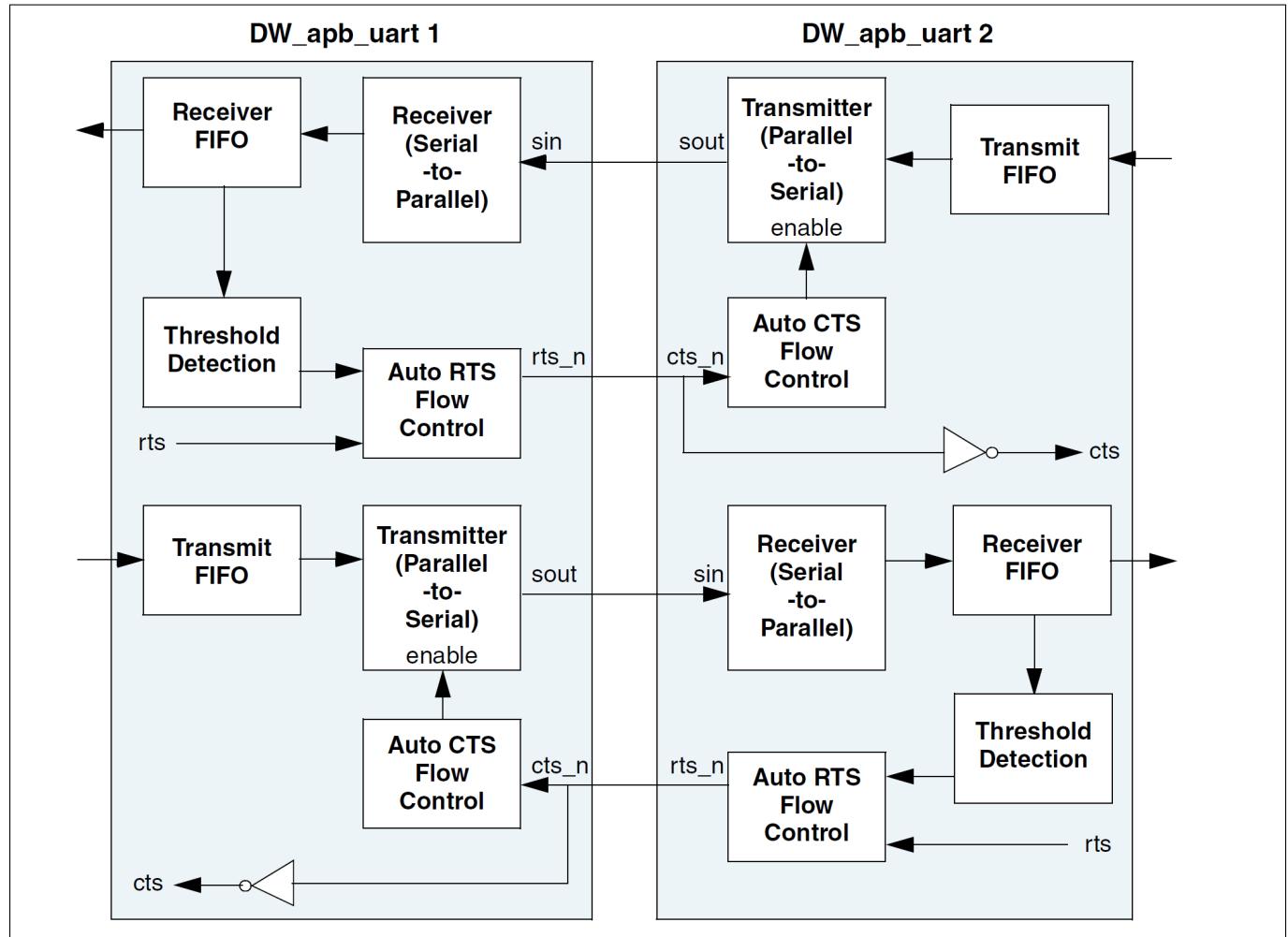


Figure 20.21: Auto Flow Control Block Diagram

- FIFOs are implemented
- RTS (MCR[1] bit and MCR[5]bit are both set)
- FIFOs are enabled (FCR[0]) bit is set)
- SIR mode is disabled (MCR[6] bit is not set)

When Auto RTS is enabled, the rts\_n output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6], but only if the RTC flow-control trigger is disabled. Otherwise, the rts\_n output is forced inactive (high) when the FIFO is almost full, where “almost full” refers to two available slots in the FIFO. When rts\_n is connected to the cts\_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space; that is, until it is completely empty.

The selectable receiver FIFO threshold values are:

- 1
- 1/4
- 1/2
- 2 less than full

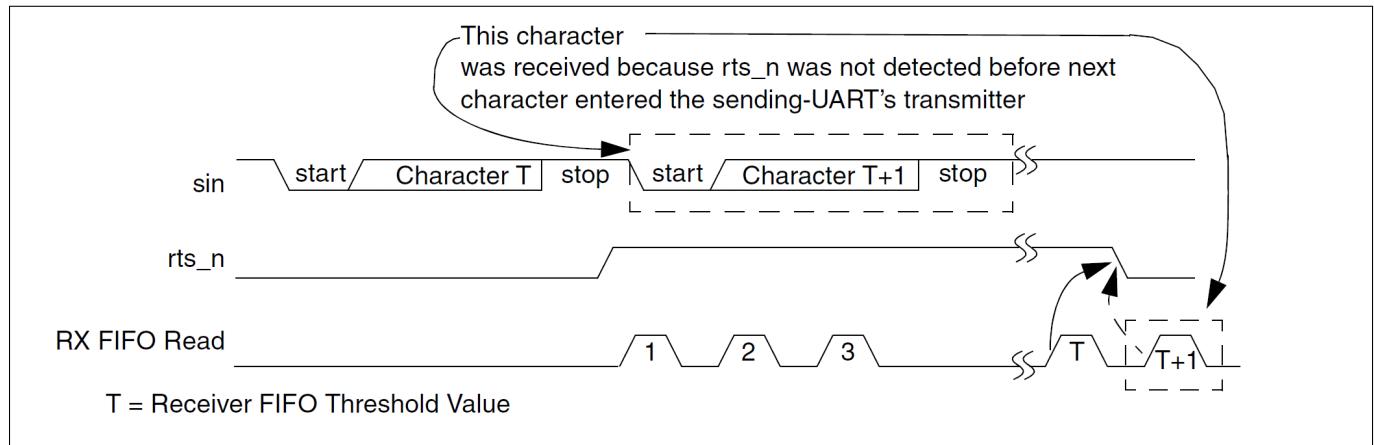
Since one additional character can be transmitted to the UART after rts\_n has become inactive—due to data already having entered the transmitter block in the other UART—setting the threshold to “2 less than full” allows maximum use of the FIFO with a safety zone of one character.

Once the receiver FIFO becomes completely empty by reading the Receiver Buffer Register (RBR), rts\_n again becomes active (low), signalling the other UART to continue sending data.

#### Note

Even if everything else is selected and the correct MCR bits are set, if the FIFOs are disabled through FCR[0] or the UART is in SIR mode (MCR[6] is set to 1), Auto Flow Control is also disabled. When Auto RTS is not implemented or disabled, rts\_n is controlled solely by MCR[1].

Figure 20.22 shows a timing diagram of the Auto RTS operation.



**Figure 20.22:** Auto RTS Timing

- Auto CTS – becomes active when the following occurs:
  - Auto Flow Control is selected during configuration
  - FIFOs are implemented

- AFCE (MCR[5] bit = 1)
- FIFOs are enabled through FIFO Control Register FCR[0] bit
- SIR mode is disabled (MCR[6] bit = 0)

When Auto CTS is enabled (active), the UART transmitter is disabled whenever the `cts_n` input becomes inactive (high); this prevents overflowing the FIFO of the receiving UART. If the `cts_n` input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed.

Therefore, when using this mode, the following happens:

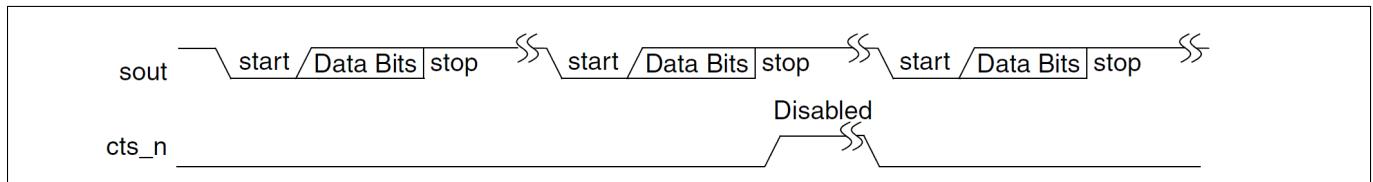
- UART status register can be read to check if transmit FIFO is full (USR[1] set to 0)
- Current FIFO level can be read using TFL register
- Programmable THRE Interrupt mode must be enabled to access “FIFO full” status using Line Status Register (LSR)

When using the “FIFO full” status, software can poll this before each write to the Transmitter FIFO; for details, refer to “Programmable THRE Interrupt”. When the `cts_n` input becomes active (low) again, transmission resumes.

#### Note

When everything else is selected, if the FIFOs are disabled using FCR[0], Auto Flow Control is also disabled. When Auto CTS is not implemented or disabled, the transmitter is unaffected by `cts_n`.

Figure 20.23 illustrates a timing diagram that shows the Auto CTS operation.



**Figure 20.23: Auto CTS Timing**

### 20.3.11 Programmable THRE Interrupt

The UART can be configured for a Programmable THRE Interrupt mode in order to increase system performance; if FIFOs are not implemented, then this mode cannot be selected.

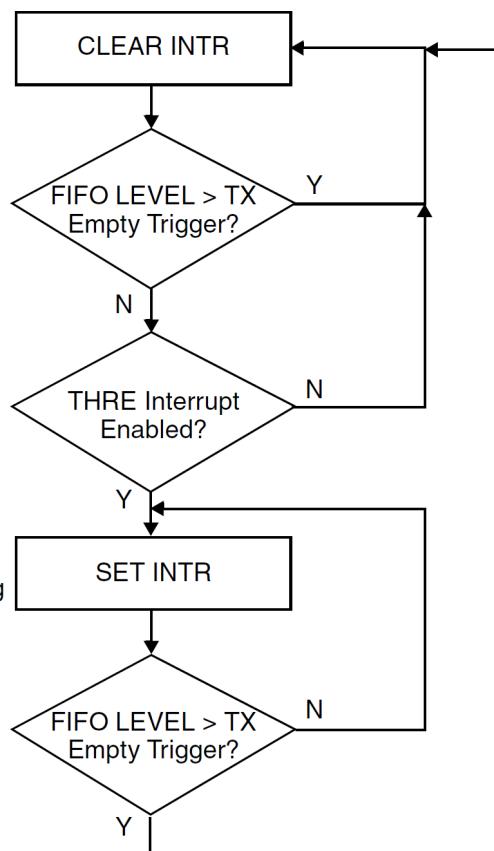
- When Programmable THRE Interrupt mode is not selected, none of the logic is implemented and the mode cannot be enabled, reducing the overall gate counts.
- When Programmable THRE Interrupt mode is selected, it can be enabled using the Interrupt Enable Register (IER[7]).

When FIFOs and THRE mode are implemented and enabled, the THRE Interrupts and `dma_tx_req_n` are active at, and below, a programmed transmitter FIFO empty threshold level, as opposed to empty, as shown in the flowchart in Figure 20.24.

The threshold level is programmed into FCR[5:4]. Available empty thresholds are:

For the THRE interrupt to be controlled as shown here, the following must be true:

- FIFO\_MODE != NONE
- THRE\_MODE = Enabled
- FIFOs enabled (FCR[0] = 1)
- THRE mode enabled (IER[7] = 1)



Under the condition that there are no other pending interrupts, the interrupt signal (intr) is asserted

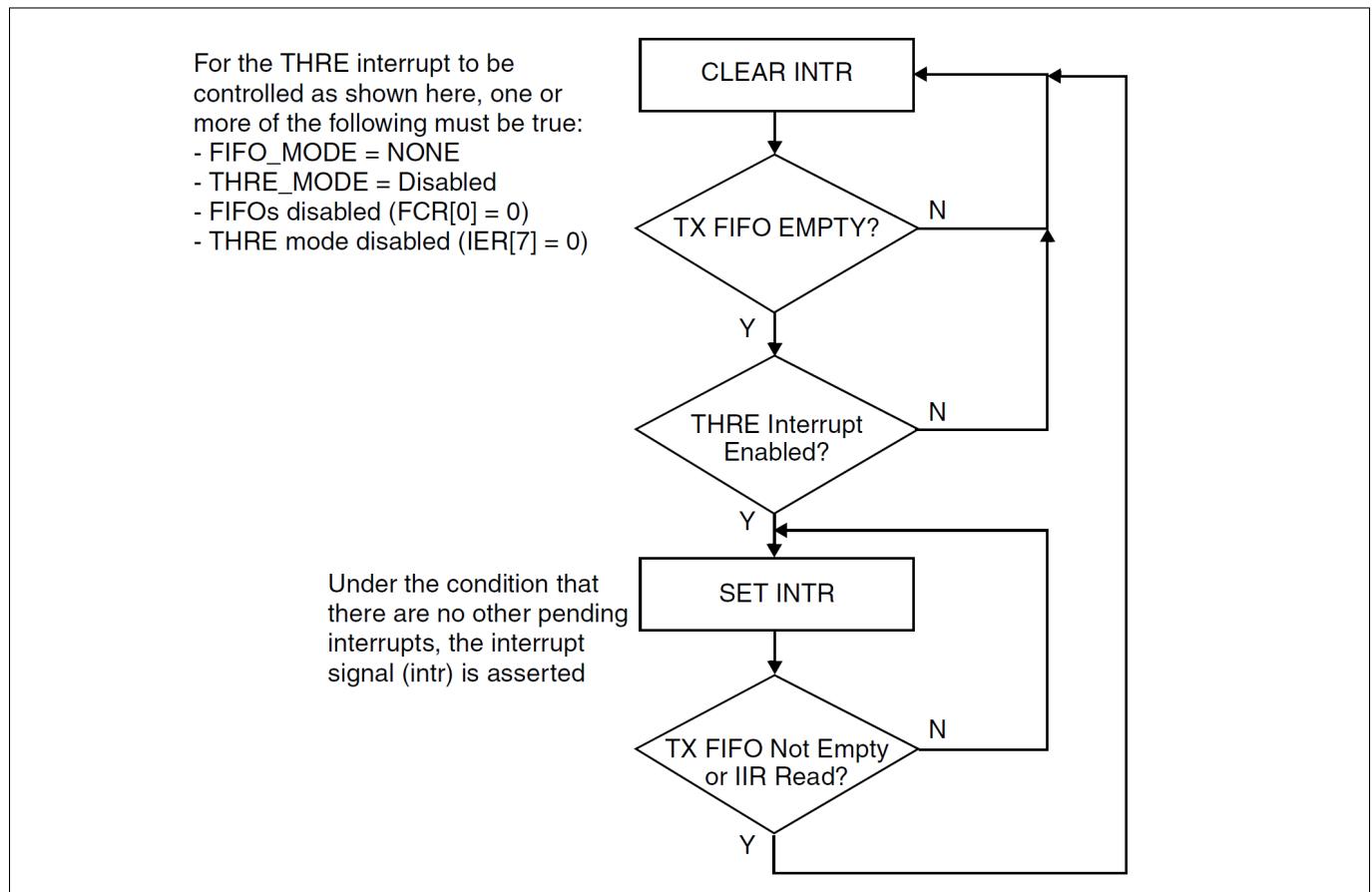
Figure 20.24: Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode

- empty
- 2
- 1/4
- 1/2

Selection of the best threshold value depends on the system's ability to begin a new transmission sequence in a timely manner. However, one of these thresholds should be optimal for increasing system performance by preventing the transmitter FIFO from running empty. For threshold setting details, refer to "FCR".

In addition to the interrupt change, the Line Status Register (LSR[5]) also switches from indicating that the transmitter FIFO is empty to the FIFO being full. This allows software to fill the FIFO for each transmit sequence by polling LSR[5] before writing another character. The flow then allows the transmitter FIFO to be filled whenever an interrupt occurs and there is data to transmit, rather than waiting until the FIFO is completely empty. Waiting until the FIFO is empty causes a reduction in performance whenever the system is too busy to respond immediately. Further system efficiency is achieved when this mode is enabled in combination with Auto Flow Control.

Even if everything else is selected and enabled, if the FIFOs are disabled using the FCR[0] bit, the Programmable THRE Interrupt mode is also disabled. When not selected or disabled, THRE interrupts and the LSR[5] bit function normally, signifying an empty THR or FIFO. Figure 20.25 illustrates the flowchart of THRE interrupt generation when not in programmable THRE interrupt mode.



**Figure 20.25:** Flowchart of Interrupt generation when not in Programmable THRE Interrupt Mode

### 20.3.12 Clock Gate Enable

The UART can be configured to have a clock gate enable output.

- When the clock gate enable option is not selected, no logic is implemented, which reduces the overall gate count.
- When the clock gate enable option is selected, the clock gate enable signal(s)—uart\_lp\_req\_pclk for single clock implementations or uart\_lp\_req\_pclk and uart\_lp\_req\_sclk for two clock implementations—is used to indicate the following:
  - Transmit and receive pipeline is clear (no data).
  - No activity has occurred.
  - Modem control input signals have not changed in more than one character time—the time taken to TX/RX a character—so that clocks can be gated. A character is made up of:

start\_bit + data\_bits + parity (optional) + stop\_bit(s))

The assertion of clock gate enable signals is an indication that the UART is inactive, so clocks may be gated in order to put the device in a low-power (lp) mode. Therefore, the following must be true for at least one character time for the assertion of the clock gate enable signal(s) to occur:

- No data in the RBR (in non-FIFO mode) or the RX FIFO is empty (in FIFO mode)
- No data in the THR (in non-FIFO mode) or the TX FIFO is empty (in FIFO mode)
- sin/sir\_in and sout/sir\_out\_n are inactive (sin/sir\_in are kept high and sout is high or sir\_out\_n is low) indicating no activity
- No change on the modem control input signals

Note, the clock gate enable assertion does not occur in the following modes of operation:

- Loopback mode
- FIFO access mode
- When transmitting a break

For example, assume a UART that is configured to have a single clock (pclk) and is programmed to transmit and receive characters of 7 bits (1 start bit, 5 data bits and 1 stop bit) and the baud clock divisor is set to 1. Therefore, the uart\_lp\_req\_pclk signal is asserted if the transmit and receive pipeline is clear, no activity has occurred and the modem control input signals have not changed for 112 (7  $\times$  16) pclk cycles.

Figure 20.26 illustrates this example.

When the assertion criteria are no longer met, the clock gate enable signal(s) are de-asserted and the clock(s) is resumed under any of these conditions:

- Either sin signal or sir\_in signal goes low
- Write to any of registers is performed
- Modem control input signals have changed when UART is in low-power (sleep) mode

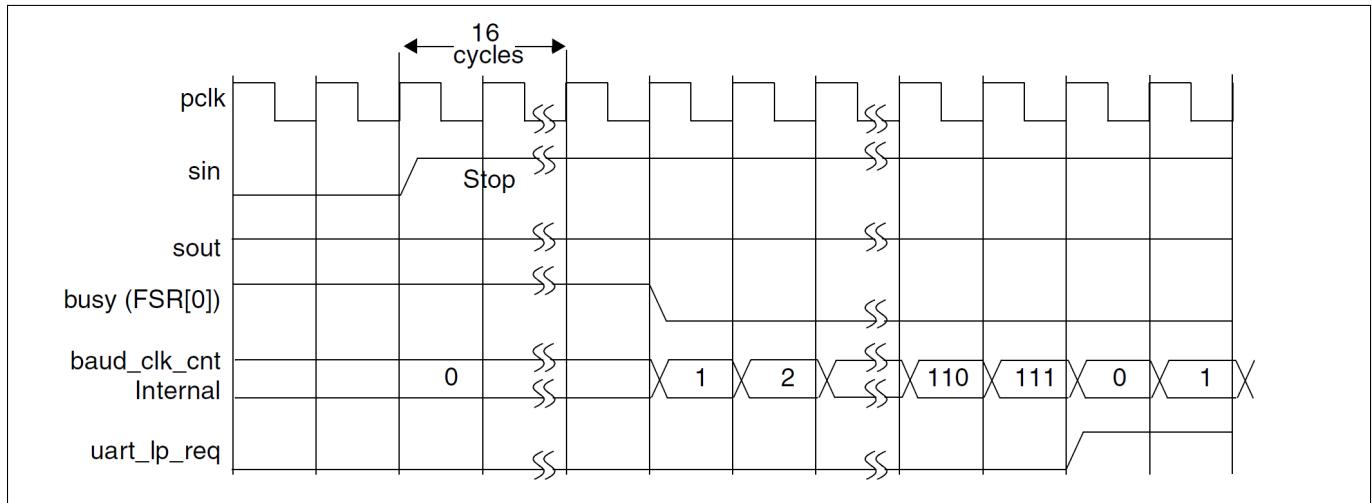


Figure 20.26: Clock Gate Enable Timing

The clock gate enable signals are de-asserted asynchronously on arrival of above mentioned events because a clock is not available to synchronize the events. Therefore, user can decide to include 2-flop syncs externally before the clock gate cell to avoid metastability issues for clock-gate latch.

#### Note

A read to any register does not de-assert the clock gate enable signal(s). The pclk clock needs to be enabled to read any of the registers in low-power mode.

The time taken for the clock(s) to resume is important in preventing receive data synchronization problems, due to the UART RX block sampling:

1. At mid-point of each bit period—after approximately 8 baud clocks—in UART (RS323) mode.
2. After that, every 16 baud clocks for a baud divisor of 1 that is 16 sclks; for a single clock implementation, this is 16 pclsks.

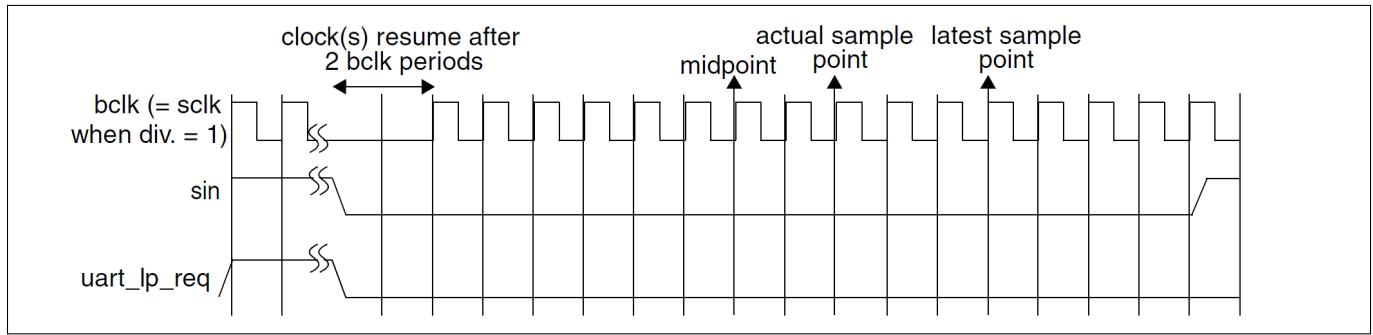
Thus, if eight or more sclk periods pass before the serial clock starts up again, the UART can get out of synchronization with the serial data it is receiving; that is, the receiver can sample into the second bit period, and if it is still 0, the receiver uses this as the start bit, and so on.

In order to avoid this problem, the clock should be resumed within five clock periods of the baud clock, which is the same as sclk if the baud divisor is set to 1; this is worst-case. If the divisor is greater, it gives a greater number of sclk cycles available before the clock must resume. This means a sample point at the 13 baud clock (at the latest) out of the 16 that are transmitted for each bit period of the character in non-SIR mode.

Figure 20.27 shows the timing diagram that illustrates the previous scenario.

This synchronization problem is magnified in SIR mode because the pulse width is only 3/16 of a bit period—three baud clocks, which for a divisor of 1 is three sclks; thus, the pulse can be missed completely. The clocks must resume before three baud clock periods elapse. However, if the first character received while in sleep mode is used only for wake-up reasons and the actual character value is unimportant, this may not become a problem.

When the UART is configured to have two clocks, if the timing of the received signal is not affected by the synchronization problem, then the minimum time to receive a character—if the baud divisor is 1—is 112 sclks:



**Figure 20.27:** Resuming Clock(s) After Low Power Mode Timing

$$1 \text{ start\_bit} + 5 \text{ data\_bits} + 1 \text{ stop\_bit} = 7 \times 16 = 112$$

Therefore, the pclk must be available before 112 sclk cycles pass in order for the received character to be synchronized to the pclk domain and stored in the RBR (in non-FIFO mode) or the RX FIFO (in FIFO mode).

### 20.3.13 DMA Support

The UART supports DMA signalling with the use of the `dma_tx_req_n` and `dma_rx_req_n` output signals to indicate:

- When data can be read
- When transmit FIFO is empty

For more information on the `dma_tx_req_n` and `dma_rx_req_n` signals, refer to “Handshaking Interface Operation”.

Note

The reset value of the `dma_tx_req_n` signal is based on the `DMA_HS_REQ_ON_RESET` parameter value.

- If `DMA_HS_REQ_ON_RESET=1`, `dma_tx_req_n` is asserted after a reset.
- If `DMA_HS_REQ_ON_RESET=0`, `dma_tx_req_n` is not asserted upon reset. It is asserted only after the LCR register is written.
- The reset under consideration is both a hardware reset (presetn) and a soft reset (by writing to the SRR register).

#### 20.3.13.1 DMA Modes

The UART uses two DMA channels—one for transmit data and one for receive data. There are two DMA modes:

- mode 0 – bit 3 of FIFO Control Register set to 0
- mode 1 – bit 3 of FIFO Control Register set to 1

Note

Only DMA mode 0 is available when FIFOs are not implemented or disabled.

### 20.3.13.1.1 DMA Mode 0

DMA mode 0 supports single DMA data transfers at a time.

In mode 0, the `dma_tx_req_n` signal:

- Goes active-low under the following conditions:
  - When Transmitter Holding Register is empty in non-FIFO mode
  - When transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled
  - When transmitter FIFO is at or below programmed threshold with Programmable THRE interrupt mode enabled
- Goes inactive when:
  - Single character has been written into Transmitter Holding Register or transmitter FIFO with Programmable THRE interrupt mode disabled
  - Transmitter FIFO is above threshold with Programmable THRE interrupt mode enabledIn mode 0, the `dma_rx_req_n` signal:
  - Goes active-low when single character is available in Receiver FIFO or Receive Buffer Register
  - Goes inactive when Receive Buffer Register or Receiver FIFO are empty, depending on FIFO mode

### 20.3.13.1.2 DMA Mode 1

DMA mode 1 supports multi-DMA data transfers, where multiple transfers are made continuously until the receiver FIFO has been emptied or the transmit FIFO has been filled.

In mode 1, the `dma_tx_req_n` signal is asserted:

- When transmitter FIFO is empty with Programmable THRE interrupt mode disabled
- When transmitter FIFO is at or below programmed threshold with Programmable THRE interrupt mode enabled

In mode 1, the `dma_tx_req_n` signal is de-asserted when the transmitter FIFO is completely full.

In mode 1, the `dma_rx_req_n` signal is asserted:

- When Receiver FIFO is at or above programmed trigger level
- When character timeout has occurred; ERBFI does not need to be set

In mode 1, the `dma_rx_req_n` signal is de-asserted when the receiver FIFO becomes empty.

### 20.3.13.1.3 Additional DMA Interface

If required for a DMA controller—such as the DW\_ahb\_dmac—you can use the DMA\_EXTRA parameter to configure the UART for additional DMA interface signals. In this case, asserting the fixed DMA signals—`dma_tx_req_n` and `dma_rx_req_n`—is similar to what was detailed in DMA Mode 0 and DMA Mode 1.

When configured for additional DMA signals, the `dma_tx_req_n` signal is asserted under the following conditions:

- When the Transmitter Holding Register is empty in non-FIFO mode
- When the transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled

- When the transmitter FIFO is at, or below the programmed threshold with Programmable THRE interrupt mode enabled.

When configured for additional DMA signals, the `dma_rx_req_n` signal is asserted under the following conditions:

- When a single character is available in Receive Buffer Register in non-FIFO mode
- When Receiver FIFO is at or above programmed trigger level in FIFO mode

With the presence of the additional handshaking signals, the UART does not have to rely on internal status and level values to recognize the completion of a request and hence remove the request. Instead, the deassertion of the DMA transmit and receive request is controlled by the assertion of the DMA transmit and receive acknowledge respectively.

When the UART is configured for additional DMA signals, responsibility of the data flow (transfer lengths) falls on the DMA (`DW_ahb_dmac`) and is controlled by the programmed burst transaction lengths. Thus, there is no need for DMA modes, and programming the FCR[3] has no effect.

#### 20.3.13.1.4 Example DMA Flow

The extra handshaking signals are explained in the following DMA flow for a UART that is configured with FIFOs and Programmable THRE interrupt mode.

As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by the UART; this is programmed into the `BLOCK_TS` field of the `CTLx` register.

The block is broken into a number of transactions, each initiated by a request from the UART. The DMA Controller must also be programmed with the number of data items (in this case, UART FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length, and is programmed into the `SRC_MSIZE/DEST_MSIZE` fields of the `DW_ahb_dmac` `CTLx` register for source and destination, respectively.

Figure 20.28 shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4.

In this case, the block size is a multiple of the burst transaction length. Therefore, the DMA block transfer consists of a series of burst transactions. If the UART makes a transmit request to this channel, four data items are written to the UART transmit FIFO. Similarly, if the UART makes a receive request to this channel, four data items are read from the UART receive FIFO. Three separate requests must be made to this DMA channel before all twelve data items are written or read.

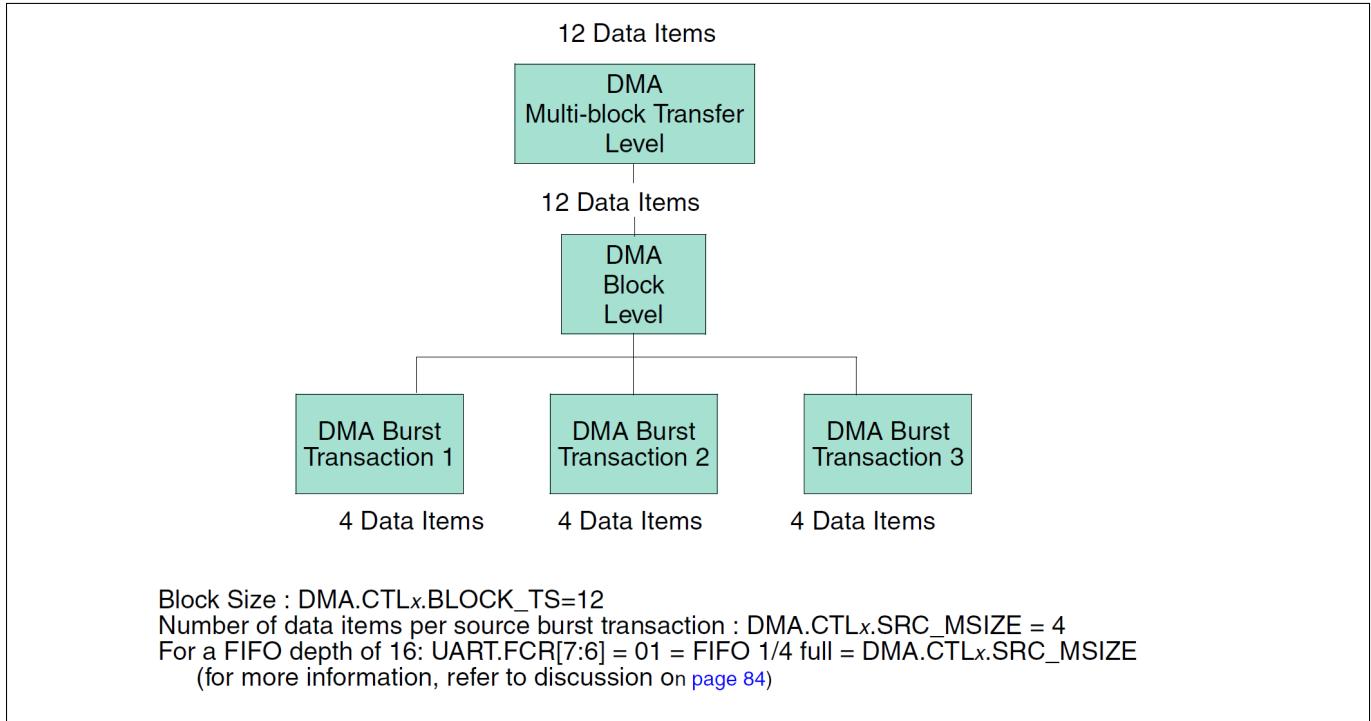
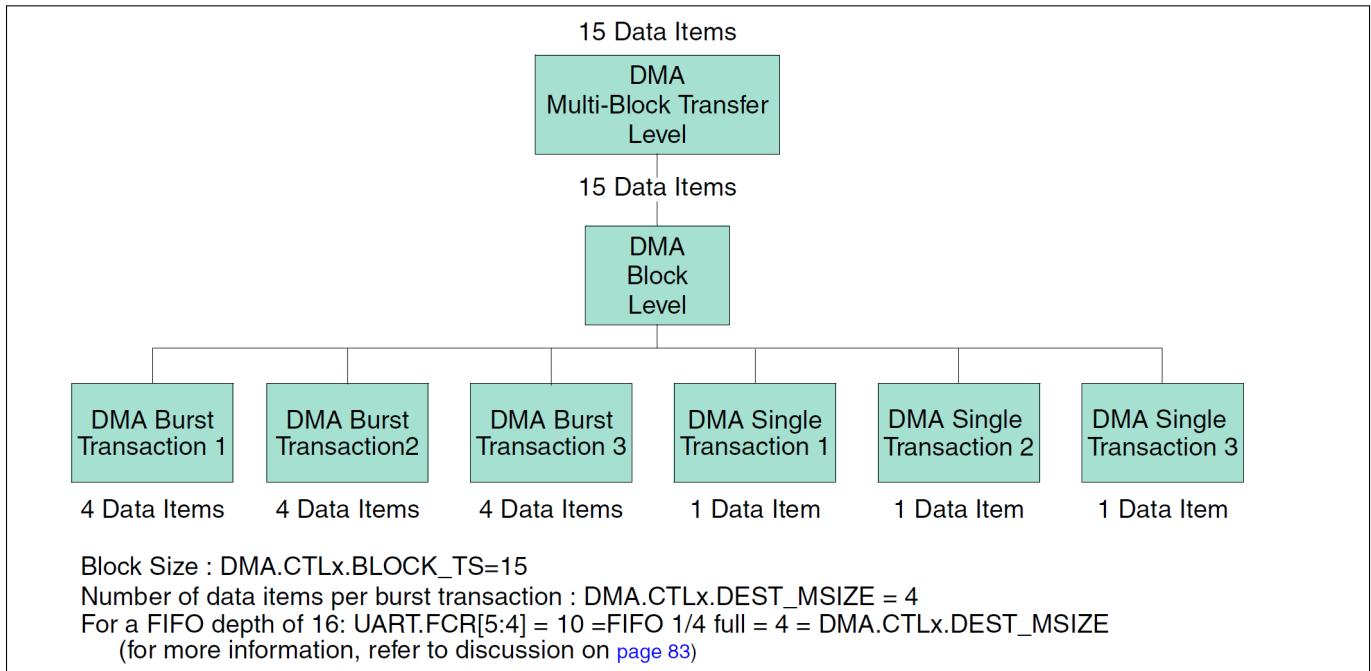
##### Note

The source and destination transfer width settings in the `DW_ahb_dmac – DMA.CTLx.SRC_TR_WIDTH` and `DMA.CTLx.DEST_TR_WIDTH` – should be set to 3'b000 because the UART FIFOs are 8 bits wide.

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in Figure 20.29, a series of burst transactions followed by single transactions are needed to complete the block transfer.

#### 20.3.13.2 Transmit Watermark Level and Transmit FIFO Underflow

During UART serial transfers, transmit FIFO requests are made to the `DW_ahb_dmac` whenever the number of entries in the transmit FIFO is less than or equal to the decoded level of the Transmit Empty Trigger (TET) of the

**Figure 20.28:** Breakdown of DMA Transfer into Burst Transactions**Figure 20.29:** Breakdown of DMA Transfer into Single and Burst Transactions

FCR register (bits 5:4); this is known as the watermark level. The DW\_ahb\_dmac responds by writing a burst of data to the transmit FIFO buffer, of length CTLx.DEST\_MSIZEx.

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty, another DMA request should be triggered. Otherwise the FIFO runs out of data (underflow). To prevent this condition, you must set the watermark level correctly.

### 20.3.13.3 Choosing Transmit Watermark Level

Consider the example where the following assumption is made:

$$\text{DMA.CTLx.DEST_MSIZE} = \text{FIFO_DEPTH} - \text{UART.FCR[5:4]}$$

The number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.

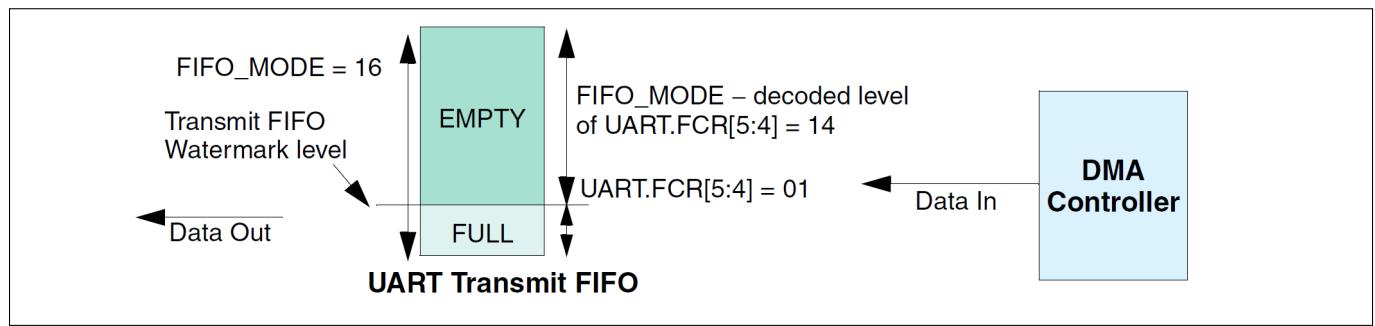


Figure 20.30: Case 1 Watermark Levels

#### 20.3.13.3.1 Case 1: FCR[5:4] = 01 — decodes to 2

- Transmit FIFO watermark level = decoded level of UART.FCR[5:4] = 2
- DMA.CTLx.DEST\_MSIZE = FIFO\_MODE – UART.FCR[5:4] = 14
- UART transmit FIFO\_MODE = 16
- DMA.CTLx.BLOCK\_TS = 56

Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

$$\text{DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE} = 56/14 = 4$$

The number of burst transactions in the DMA block transfer is 4., but the watermark level—decoded level of UART.FCR[5:4]—is quite low. Therefore, the probability of a UART underflow is high where the UART serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.

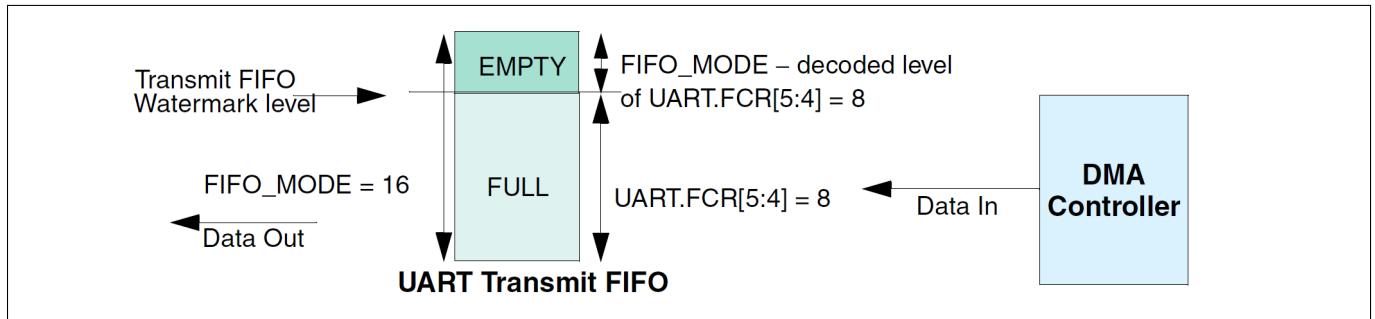


Figure 20.31: Case 2 Watermark Levels

#### 20.3.13.3.2 Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8)

- Transmit FIFO watermark level = decoded level of UART.FCR[5:4] = 8
- DMA.CTLx.DEST\_MSIZE = FIFO\_MODE - UART.FCR[5:4] = 8
- UART transmit FIFO\_MODE = 16
- DMA.CTLx.BLOCK\_TS = 56

Number of burst transactions in Block:

$$\text{DMA.CTLx.BLOCK_TS}/\text{DMA.CTLx.DEST_MSIZE} = 56/8 = 7$$

In this block transfer, there are seven destination burst transactions in a DMA block transfer, but the watermark level—decoded level of UART.FCR[5:4]—is high. Therefore, the probability of a UART underflow is low because the DMA controller has enough time to service the destination burst transaction request before the UART transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of AMBA bursts per block and worse bus utilization than Case 1.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of:

$$\text{rate of UART data transmission} : \text{rate of DMA response to destination burst requests}$$

For example, both of the following increases the rate at which the DMA controller can respond to burst transaction requests:

- Promoting channel to highest priority channel in DMA
- Promoting DMA master interface to highest priority master in AMBA layer

This in turn enables the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

#### 20.3.13.4 Selecting DEST\_MSIZEx and Transmit FIFO Overflow

As can be seen from Figure 20.31, programming DMA.CTLx.DEST\_MSIZEx to a value greater than the watermark level that triggers the DMA request can cause overflow when there is not enough space in the UART transmit FIFO to service the destination burst request. Therefore, use the following in order to avoid overflow:

$$\text{DMA.CTLx.DEST_MSIZEx} \leq \text{UART.FIFO_DEPTH} - \text{decoded level of UART.FCR[5:4]} \quad (1)$$

In Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8), the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST\_MSIZEx. Thus, the transmit FIFO can be full, but not overflowed, at the completion of the burst transaction.

Therefore, for optimal operation, DMA.CTLx.DEST\_MSIZEx should be set at the FIFO level that triggers a transmit DMA request; that is:

$$\text{DMA.CTLx.DEST_MSIZEx} = \text{UART.FIFO_DEPTH} - \text{decoded level of UART.FCR[5:4]} \quad (2)$$

This is the setting used in Figure 20.29.

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, which in turn improves AMBA bus utilization.

##### Note

The transmit FIFO is not full at the end of a DMA burst transfer if the UART has successfully transmitted one data item or more on the UART serial transmit line during the transfer.

#### 20.3.13.5 Receive Watermark Level and Receive FIFO Overflow

During UART serial transfers, receive FIFO requests are made to the DW\_ahb\_dmac whenever the number of entries in the receive FIFO is at or above the decoded level of Receiver Trigger (RT) of the FCR[7:6]. This is known as the watermark level. The DW\_ahb\_dmac responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC\_MSIZEx.

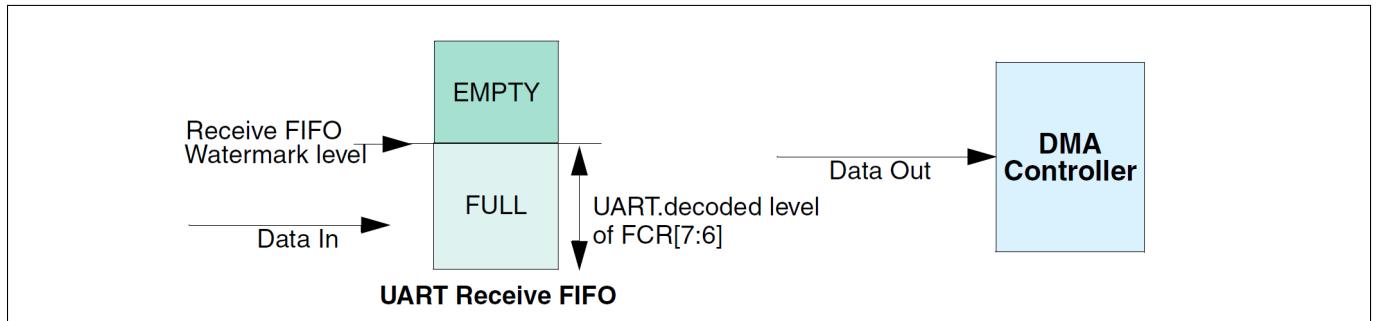
Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO fills with data (overflow). To prevent this condition, you must correctly set the watermark level.

#### 20.3.13.6 Choosing the Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level—decoded level of FCR[7:6]—should be set to minimize the probability of overflow. It is a trade-off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

#### 20.3.13.7 Selecting SRC\_MSIZEx and Receive FIFO Underflow

As can be seen in Figure 20.32, programming a source burst transaction length greater than the watermark level can cause underflow when there is not enough data to service the source burst request. Therefore, equation (3) below must be adhered to in order to avoid underflow.



**Figure 20.32:** UART Receive FIFO

If the number of data items in the receive FIFO is equal to the source burst length at the time the burst request is made – DMA.CTLx.SRC\_MSIZE – the receive FIFO can be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC\_MSIZE should be set at the watermark level; that is:

$$\text{DMA.CTLx.SRC_MSIZE} = \text{decoded level of FCR[7:6]} \quad (3)$$

Adhering to equation (3) reduces the number of DMA bursts in a block transfer, and this in turn can improve AMBA bus utilization.

#### Note

The receive FIFO is not empty at the end of the source burst transaction if the UART has successfully received one data item or more on the UART serial receive line during the burst.

#### 20.3.13.8 Handshaking Interface Operation

- `dma_tx_req_n`, `dma_rx_req_n` – The request signals for source and destination—`dma_tx_req_n` and `dma_rx_req_n`—are activated when their corresponding FIFOs reach the watermark levels.

The DW\_ahb\_dmac uses edge detection of the `dma_tx_req_n` signal/`dma_rx_req_n` to identify a request on the channel. Upon reception of the `dma_tx_ack_n`/`dma_rx_ack_n` signal from the DW\_ahb\_dmac to indicate the burst transaction is complete, the UART de-asserts the burst request signals—`dma_tx_req_n`/`dma_rx_req_n`—until `dma_tx_ack_n`/`dma_rx_ack_n` is de-asserted by the DW\_ahb\_dmac.

When the UART samples that `dma_tx_ack_n`/`dma_rx_ack_n` is de-asserted, it can re-assert the `dma_tx_req_n`/`dma_rx_req_n` of the request line if their corresponding FIFOs exceed their watermark levels—back-to-back burst transaction. If this is not the case, the DMA request lines remain de-asserted.

Figure 20.33 shows a timing diagram of a burst transaction where `pclk` = `hclk`.

Figure 20.34 shows two back-to-back burst transactions where the `hclk` frequency is twice the `pclk` frequency.

The handshaking loop is as follows:

- a. `dma_tx_req_n`/`dma_rx_req_n` asserted by UART
- b. `dma_tx_ack_n`/`dma_rx_ack_n` asserted by DW\_ahb\_dmac
- c. `dma_tx_req_n`/`dma_rx_req_n` de-asserted by UART
- d. `dma_tx_ack_n`/`dma_rx_ack_n` de-asserted by DW\_ahb\_dmac
- e. `dma_tx_req_n`/`dma_rx_req_n` re-asserted by UART, if back-to-back transaction is required

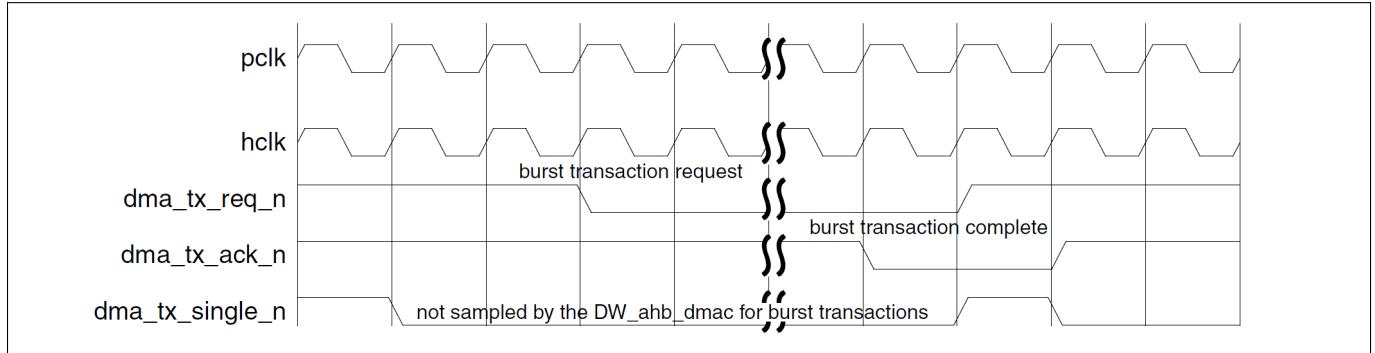


Figure 20.33: Burst Transaction –  $pclk = hclk$

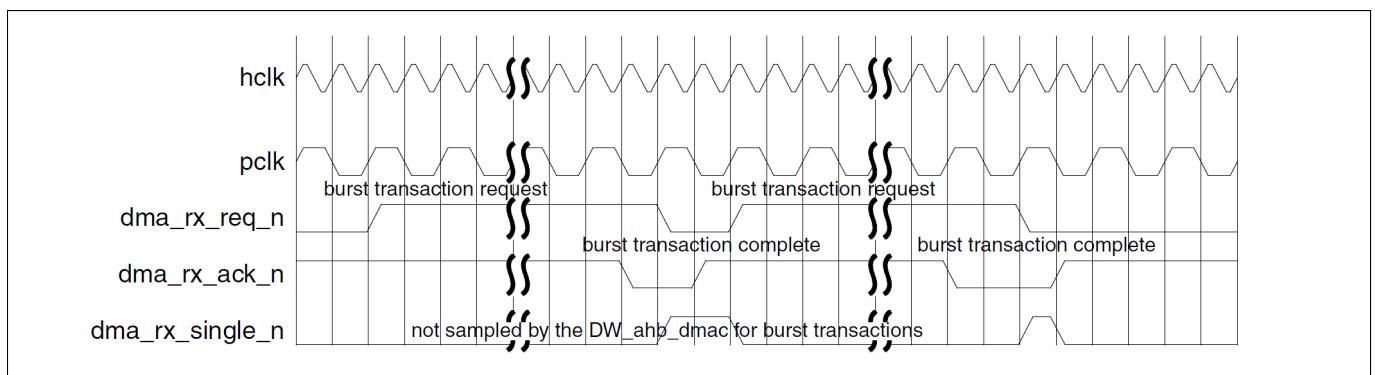


Figure 20.34: Back-to-Back Burst Transactions –  $hclk = 2*pclk$

### Note

The burst transaction request signals, `dma_tx_req_n` and `dma_rx_req_n`, are generated in the UART off `pclk` and sampled in the `DW_ahb_dmac` by `hclk`. The acknowledge signals, `dma_tx_ack_n` and `dma_rx_ack_n`, are generated in the `DW_ahb_dmac` off `hclk` and sampled in the UART of `pclk`. The handshaking mechanism between the `DW_ahb_dmac` and the UART supports quasi-synchronous clocks; that is, `hclk` and `pclk` must be phasealigned, and the `hclk` frequency must be a multiple of the `pclk` frequency.

- Note the following:
  - \* Once asserted, the burst request lines—`dma_tx_req_n/dma_rx_req_n`—remain asserted until their corresponding `dma_tx_ack_n/dma_rx_ack_n` signal is received, even if the respective FIFOs drop below their watermark levels during the burst transaction.
  - \* The `dma_tx_req_n/dma_rx_req_n` signals are de-asserted when their corresponding `dma_tx_ack_n /dma_rx_ack_n` signals are asserted, even if the respective FIFOs exceed their watermark levels.
- `dma_tx_single_n, dma_rx_single_n`
  - `dma_tx_single_n` – status signal that is asserted when there is at least one free entry in the transmit FIFO; it is cleared when the transmit FIFO is full.
  - `dma_rx_single_n` – status signal that is asserted when there is at least one valid data entry in the receive FIFO; it is cleared when the receive FIFO is empty.

These signals are needed by only the `DW_ahb_dmac` for the case where the block size—`CTLx.BLOCK_TS`—that is programmed into the `DW_ahb_dmac` is not a multiple of the burst transaction length—`CTLx.SRC_MSIZEx`, `CTLx.DEST_MSIZEx`—shown in Figure 20.29. In this case, the DMA single outputs inform the `DW_ahb_dmac` that it is still possible to perform single data item transfers, so it can access all data items in the transmit/receive FIFO and complete the DMA block transfer. Otherwise, the DMA single outputs from the UART are not sampled by the `DW_ahb_dmac`.

This is illustrated in the following example.

Receive FIFO Channel of the UART:

$$\text{DMA.CTLx.SRC_MSIZE} = \text{decoded level of UART.FCR[7:6]} = 4$$

$$\text{DMA.CTLx.BLOCK_TS} = 12$$

Block transfer:

$$\text{DMA.CTLx.SRC_MSIZE} = \text{decoded level of UART.FCR[7:6]} = 4$$

$$\text{DMA.CTLx.BLOCK_TS} = 15$$

For the example in Figure 20.28, with the block size set to 12, the `dma_rx_req_n` signal is asserted when four data items are present in the receive FIFO. The `dma_rx_req_n` signal is asserted three times during the UART serial transfer, ensuring that all 12 data items are read by the `DW_ahb_dmac`. All DMA requests read a block of data items and no single DMA transactions are required. The block transfer is made up of three burst transactions.

The first 12 data items are transferred using three burst transactions. But when the last three data frames enter the receive FIFO, the `dma_rx_req_n` signal is not activated because the FIFO level is below the watermark level. The `DW_ahb_dmac` samples `dma_rx_single_n` and completes the DMA block transfer using three single transactions. The block transfer is made up of three burst transactions, followed by three single transactions.

Figure 20.35 shows a single transaction. The handshaking loop is as follows:

- a. dma\_tx\_single\_n/dma\_rx\_single\_n asserted by UART
- b. dma\_tx\_ack\_n/dma\_rx\_ack\_n asserted by DW\_ahb\_dmac
- c. dma\_tx\_single\_n/dma\_rx\_single\_n de-asserted by UART
- d. dma\_tx\_ack\_n/dma\_rx\_ack\_n de-asserted by DW\_ahb\_dmac

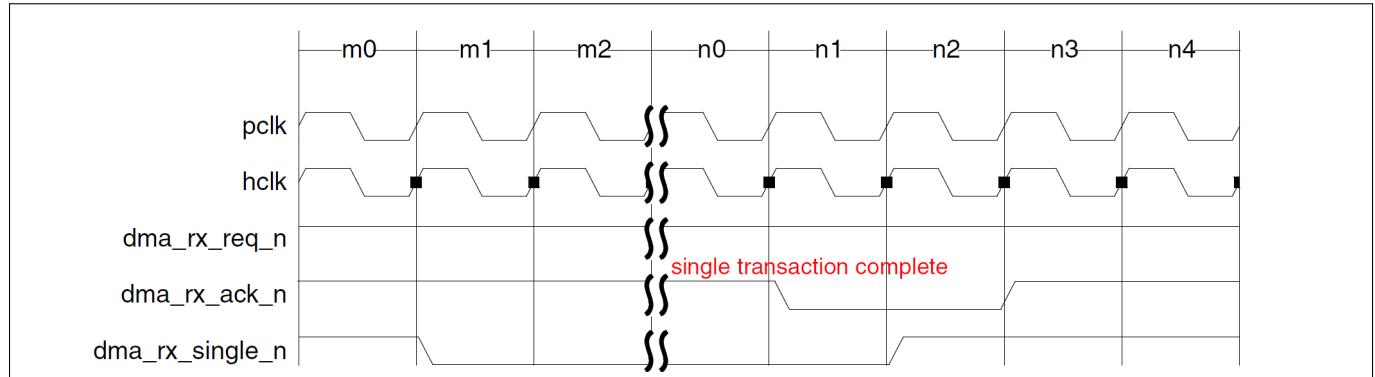


Figure 20.35: Single Transaction

Figure 20.36 shows a burst transaction, followed by three back-to-back single transactions, where the hclk frequency is twice the pclk frequency.

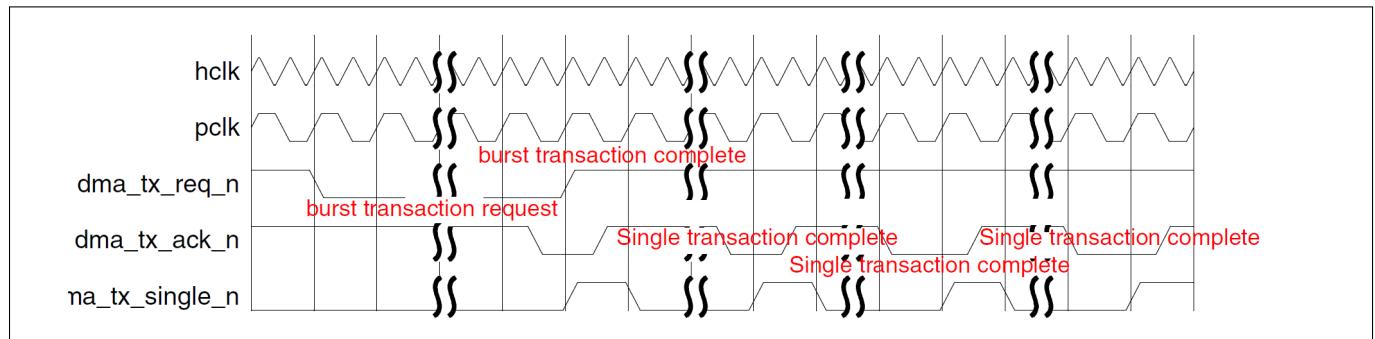


Figure 20.36: Burst Transaction + 3 Back-to-Back Singles – hclk = 2\*pclk

#### Note

The single transaction request signals, `dma_tx_single_n` and `dma_rx_single_n`, are generated in the UART on the `pclk` edge and sampled in `DW_ahb_dmac` on `hclk`. The acknowledge signals, `dma_tx_ack_n` and `dma_rx_ack_n`, are generated in the `DW_ahb_dmac` on the `hclk` edge `hclk` and sampled in the UART on `pclk`. The handshaking mechanism between the `DW_ahb_dmac` and the UART supports quasynchronous clocks; that is, `hclk` and `pclk` must be phase aligned and the `hclk` frequency must be a multiple of `pclk` frequency.

#### 20.3.13.9 Potential Deadlock Conditions in UART/DW\_ahb\_dmac Systems

There is a risk of a deadlock occurring if both of the following are true:

- DW\_ahb\_dmac is used to access UART FIFOs
- DMA burst transaction length is set to value smaller than or equal to UART Rx FIFO threshold
- When UART is used in DMA mode 1 with auto-flow control mode enabled

**20.3.13.9.1 Deadlock When DMA Burst Transaction Length Smaller Than Rx FIFO Threshold** When operating in autoflow control mode with the RTC flow trigger threshold disabled, the UART de-asserts rts\_n when the Rx FIFO threshold is reached, and it asserts it again when the Rx FIFO is empty. At the same time, the UART asserts dma\_rx\_req\_n, requesting a burst transaction from the DW\_ahb\_dmac.

If the DMA burst transaction length is equal to or greater than the Rx FIFO threshold, the DW\_ahb\_dmac reads from the Rx FIFO until it is empty, causing rts\_n to be re-asserted. This in turn allows more data to be received by the UART and the Rx FIFO to fill again.

However, if the DW\_ahb\_dmac burst transaction length is smaller than the UART Rx FIFO threshold, some data is left in the UART Rx FIFO after completion of the burst transaction. This prevents the rts\_n signal from being asserted.

Because the amount of data in the Rx FIFO is below the threshold, the UART asserts the dma\_rx\_single\_n signal—instead of dma\_rx\_req\_n—requesting a DMA single transaction from the

DW\_ahb\_dmac. However, unless it is operating in the single transaction region, the DW\_ahb\_dmac ignores single transaction requests.

A deadlock condition is then reached:

- UART does not receive any extra characters because the rts\_n signal is de-asserted; no data can be pushed into the Rx FIFO to fill it up to the threshold level again and generate a new burst transaction request from the DW\_ahb\_dmac; only single transaction requests can be generated.
- Unless it has reached the single transaction region, the DW\_ahb\_dmac ignores single transaction requests and does not read from the Rx FIFO; the Rx FIFO cannot be emptied, which prevents the rts\_n signal from being asserted again

Table 20.4 illustrates this condition.

DW_apb_uart Settings	DW_ahb_dmac Settings
Component configured for 32 byte deep Rx FIFO	Block size set to 100 bytes (BLOCK_TS=100)
Autoflow mode enabled (AFCE=1)	Source transaction width set to 1 byte (SRC_TR_WIDTH=1)
Rx FIFO threshold set to $\frac{1}{2}$ full; that is, 30 bytes (RCVR=11)	Source burst transaction length set to 16 (SRC_MSIZE=16)

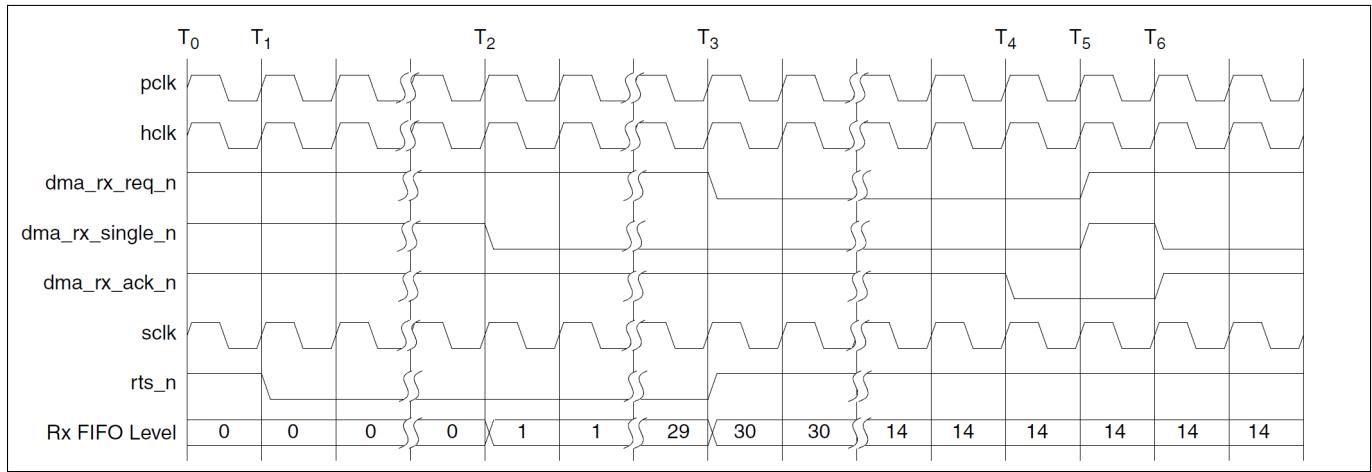
**Table 20.4:** DW\_apb\_uart/DW\_ahb\_dmac Settings for Deadlock When Transaction Less Than Rx FIFO Threshold

The timing diagram in Figure 20.37 illustrates the sequence of events that lead to this deadlock condition.

Note

For the sake of simplicity, pclk, hclk and sclk are shown to be identical; however, this is not a constraint for the occurrence of deadlock. Additionally, in the interest of simplicity, some events are represented as taking place simultaneously; however, in reality this might not be strictly the case and these events can be separated by a small number of clock cycles.

In Figure 20.37, the following events are shown:



**Figure 20.37:** Example of UART and DW\_ahb\_dmac Deadlock Occurrence

- T1 – The UART is programmed and enabled; rts\_n is asserted to initiate the reception of characters.
- T2 – The first character is received by the UART and pushed into the RxFIFO; dma\_rx\_single\_n is asserted as a consequence, but because DW\_ahb\_dmac is not in the single transfer region, this request is ignored.
- T3 – The 30th character is received and pushed into the Rx FIFO. As a consequence:
  - rts\_n is de-asserted, stopping any further characters from being received
  - dma\_rx\_req\_n signal is asserted
  - DW\_ahb\_dmac attends this request and starts reading data from Rx FIFO
- T4 – The 16th character popped from the Rx FIFO is received by the DW\_ahb\_dma, which asserts dma\_rx\_ack\_n to signal the completion of the DMA burst transaction. Since the DMA burst transaction size is set to 16 and the Rx FIFO threshold is set to 30, there are fourteen characters left in the Rx FIFO after the DMA burst transaction completes.
- T5 – One cycle after dma\_rx\_ack\_n is asserted, the UART de-asserts dma\_rx\_req\_n and dma\_rx\_single\_n as part of the DMA handshaking protocol.
- T6 – One cycle after dma\_rx\_req\_n is de-asserted, the DW\_ahb\_dmac de-asserts dma\_rx\_ack\_n to complete the DMA handshaking protocol. At the same time, the UART re-asserts dma\_req\_single\_n because there are fourteen characters in the Rx FIFO. For the same reason, rts\_n is kept de-asserted; since the DW\_ahb\_dmac is not in the single transfer region, it ignores the single transaction request and a deadlock is created.

This deadlock condition can be avoided if:

- The Rx FIFO threshold level is set to a value equal to or smaller than the DMA burst transaction size. This ensures the Rx FIFO is always empty after a DMA burst transaction completes and rts\_n is asserted accordingly.
- The DMA block size is set to a value smaller than twice the DMA burst transaction length. This guarantees the DW\_ahb\_dmac enters the single transaction region after the DMA burst transaction completes. It then accepts single transaction requests from the UART, allowing the Rx FIFO to be emptied. In this case, the DMA burst size can be configured to be smaller than the Rx FIFO threshold level.

**20.3.13.9.2 Deadlock When DMA Burst Transaction Length Equal To Rx FIFO Threshold** If the DMA burst transaction length is identical to the UART Rx FIFO threshold, there is risk of a deadlock condition occurring when a character is received after rts\_n is de-asserted.

The UART de-asserts rts\_n when the Rx FIFO threshold is reached. However, it is possible the component at the other end of the line starts transmitting a new character before it detects the de-assertion of its cts\_n input. When this happens, the character transmission completes normally, which means an extra character will be received and pushed into the Rx FIFO (unless it is already full).

At the same time that rts\_n is de-asserted, the UART asserts dma\_rx\_req, requesting a DMA burst transaction from the DW\_ahb\_dmac. After the DW\_ahb\_dmac completes this burst transaction—with length equal to the Rx FIFO threshold—there is one character left in the Rx FIFO, preventing rts\_n from being asserted again.

The UART asserts the dma\_rx\_single\_n signal—instead of dma\_rx\_req\_n—requesting a DMA single transaction from the DW\_ahb\_dmac. However, unless it is operating in the single-transaction region, the DW\_ahb\_dmac ignores single-transaction requests.

A deadlock condition is then reached:

- The UART does not receive any extra characters because the rts\_n signal is de-asserted. No data can be pushed into the Rx FIFO to fill it up to the threshold level again and generate a new burst transaction request from the DW\_ahb\_dma; only single-transaction requests can be generated.
- Unless it has reached the single-transaction region, the DW\_ahb\_dmac ignores single-transaction requests and does not read from the Rx FIFO. The Rx FIFO cannot be emptied, which prevents the rts\_n signal from being asserted again.

This deadlock condition can be avoided if:

- The Rx FIFO threshold level is set to a value smaller than the DMA burst transaction size. This ensures that the Rx FIFO is always empty after a DMA burst transaction completes, regardless of whether or not one extra character is received and rts\_n is asserted accordingly.
- The DMA block size is set to a value smaller than twice the DMA burst transaction length. This guarantees that the DW\_ahb\_dmac enters the single transaction region after the DMA burst transaction completes. It then accepts single transaction requests from the UART, allowing the Rx FIFO to be emptied.

#### Note

This deadlock condition is not expected to occur frequently under normal operating conditions. A timeout interrupt would be generated in this case, which can be used to detect the occurrence of this deadlock condition.

### 20.3.14 Reset Signals

When configured for asynchronous serial clock operation, the UART includes two separate reset signals, each dedicated to its own clock domain:

- presetn resets logic in pclk clock domain
- s\_rst\_n resets logic in sclk clock domain

In order to avoid serious operational failures, both clock domains of the UART must be reset before any attempt is made to send or receive data on the serial line; that is, it is an illegal operation to reset just one clock domain of the UART without resetting the other clock domain.

Each reset signal must be de-asserted synchronously with the corresponding clock signal.

When asserting the reset signals, the s\_rst\_n signal should be asserted before or at the same time as presetn; this prevents any unexpected activity on the serial line that might result from resetting the programming registers without resetting the serial logic.

Similarly, when de-asserting the reset signals, s\_rst\_n should be de-asserted before presetn is de-asserted. The safest procedure for resetting UART is as follows:

1. Assert s\_rst\_n and presetn; the sequence of asserting these two signals and their timing relationship with sclk and pclk are not important
2. De-assert s\_rst\_n synchronously with sclk
3. De-assert presetn synchronously with pclk

Both reset signals should be active for at least three cycles of the respective clock signal.

## 20.4 UART Register Description

### 20.4.1 Register Map Summary

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])

Register	Offset	Description	Reset Value
Receive Buffer Register.	0x00 0x00 0x00 0x00 0x00 0x00	RBR	0x0000_0000
Transmit Holding Register.	0x00 0x00 0x00 0x00 0x00 0x00	THR	0x0000_0000
divisor latch low.	0x00 0x00 0x00 0x00 0x00 0x00	DLL	0x0000_0000
interrupt enable register	0x04 0x04 0x04 0x04 0x04 0x04	IER	0x0000_0000
divisor latch high	0x04 0x04 0x04 0x04 0x04 0x04	DLH	0x0000_0000
interrupt identity register	0x08 0x08 0x08 0x08 0x08 0x08	IIR	0x0000_0001
FIFO control register	0x08 0x08 0x08 0x08 0x08 0x08	FCR	0x0000_0000
line control register	0x0C 0x0C 0x0C 0x0C 0x0C 0x0C	LCR	0x0000_0000
modem control register	0x10 0x10 0x10 0x10 0x10 0x10	MCR	0x0000_0000

line status register	0x14 0x14 0x14 0x14 0x14 0x14	LSR	0x0000_0060
modem status register	0x18 0x18 0x18 0x18 0x18 0x18	MSR	0x0000_0000
scratch register	0x1C 0x1C 0x1C 0x1C 0x1C 0x1C	SCR	0x0000_0000
Low Power Divisor Latch Low	0x20 0x20 0x20 0x20 0x20 0x20	LPDLL	0x0000_0000
Low Power Divisor Latch Low	0x24 0x24 0x24 0x24 0x24 0x24	LPDLL	0x0000_0000
shadow receive buffer registers	0x30 0x30 0x30 0x30 0x30 0x30	SRBRn	0x0000_0000
shadow transmit holding registers	0x30 0x30 0x30 0x30 0x30 0x30	STHRn	0x0000_0000
FIFO access register	0x70 0x70 0x70 0x70 0x70 0x70	FAR	0x0000_0000
transmit FIFO read	0x74 0x74 0x74 0x74 0x74 0x74	TFR	0x0000_0000
receiver FIFO write	0x78 0x78 0x78 0x78 0x78 0x78	RFW	0x0000_0000
uart status register	0x7C 0x7C 0x7C 0x7C 0x7C 0x7C	USR	0x0000_0006
transmit FIFO level	0x80 0x80 0x80 0x80 0x80 0x80	TFL	0x0000_0000

receive FIFO level	0x84 0x84 0x84 0x84 0x84 0x84	RFL	0x0000_0000
software reset register	0x88 0x88 0x88 0x88 0x88 0x88	SRR	0x0000_0000
shadow request to send	0x8C 0x8C 0x8C 0x8C 0x8C 0x8C	SRTS	0x0000_0000
shadow request to send	0x90 0x90 0x90 0x90 0x90 0x90	SRTS	0x0000_0000
shadow break control	0x94 0x94 0x94 0x94 0x94 0x94	SBCR	0x0000_0000
shadow FIFO enable	0x98 0x98 0x98 0x98 0x98 0x98	SFE	0x0000_0000
shadow receiver trigger	0x9C 0x9C 0x9C 0x9C 0x9C 0x9C	SRT	0x0000_0000
shadow transmitter trigger	0xA0 0xA0 0xA0 0xA0 0xA0 0xA0	STET	0x0000_0000
halt Tx	0xA4 0xA4 0xA4 0xA4 0xA4 0xA4	HTX	0x0000_0000
dma software acknowledge	0xA8 0xA8 0xA8 0xA8 0xA8 0xA8	DMASA	0x0000_0000
Transceiver Control Register	0xAC 0xAC 0xAC 0xAC 0xAC 0xAC	TCR	0x0000_0006
Driver Output Enable Register	0xB0 0xB0 0xB0 0xB0 0xB0 0xB0	DE_EN	0x0000_0000

Receiver Output Enable Register	0xB4 0xB4 0xB4 0xB4 0xB4 0xB4	RE_EN	0x0000_0000
Driver Output Enable Timing Register	0xB8 0xB8 0xB8 0xB8 0xB8 0xB8	DET	0x0000_0000
TurnAround Timing Register	0xBC 0xBC 0xBC 0xBC 0xBC 0xBC	TAT	0x0000_0000
Divisor Latch Fraction Register	0xC0 0xC0 0xC0 0xC0 0xC0 0xC0	DLF	0x0000_0000
Receive Address Register	0xC4 0xC4 0xC4 0xC4 0xC4 0xC4	RAR	0x0000_0000
Transmit Address Register	0xC8 0xC8 0xC8 0xC8 0xC8 0xC8	TAR	0x0000_0000
Line Extended Control Register	0xCC 0xCC 0xCC 0xCC 0xCC 0xCC	LCR_EXT	0x0000_0000
Component Parameter Register	0xF4 0xF4 0xF4 0xF4 0xF4 0xF4	CPR	0x0000_0000
UART Component Version	0xF8 0xF8 0xF8 0xF8 0xF8 0xF8	UCV	0x0000_0000
Component Type Register	0xFC 0xFC 0xFC 0xFC 0xFC 0xFC	CTR	0x4457_0110

### 20.4.2 Receive Buffer Register.

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
RBR	[8:0]	R	<p>Receive Buffer Register.</p> <p>This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to 0), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an over-run error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to 1), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost and an overrun error occurs.</p> <p>This register is activated when lcr bit7 is 0.</p>	0

### 20.4.3 Transmit Holding Register.

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
THR	[8:0]	W	<p>Transmit Holding Register.            This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.            If in non-FIFO mode or FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.            If in FIFO mode and FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that is set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.            This register is activated when lcr bit7 is 0.</p>	0

### 20.4.4 divisor latch low.

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
DLL	[7:0]	R/W	<p>Divisor Latch (Low).            This register makes up the lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the USART. The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (CLOCK_MODE == Enabled)) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor). Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.            This register is activated when lcr bit7 is 1.</p>	0

### 20.4.5 interrupt enable register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x04, 0x04, 0x04, 0x04, 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
PTIME	[7]	R/W	Programmable THRE Interrupt Mode Enable. Writeable only when THRE_MODE_USER == Enabled, always readable. This is used to enable/disable the generation of THRE Interrupt. Values: 0x0 (DISABLED): Disable Programmable THRE Interrupt Mode 0x1 (ENABLED): Enable Programmable THRE Interrupt Mode This register is activated when lcr bit7 is 0.	0
RSVD	[6:5]	-	Reserved	-
ELCOLR	[4]	R/W	Interrupt Enable Register: ELCOLR, this bit controls the method for clearing the status in the LSR register. This is applicable only for Overrun Error, Parity Error, Framing Error, and Break Interrupt status bits. 0 = LSR status bits are cleared either on reading Rx FIFO (RBR Read) or On reading LSR register. 1 = LSR status bits are cleared only on reading LSR register. Writeable only when LSR_STATUS_CLEAR == Enabled, always readable. Values: 0x0 (DISABLED): Disable ALC 0x1 (ENABLED): Enable ALC This register is activated when lcr bit7 is 0.	0
EDSSI	[3]	R/W	Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. Values: 0x0 (DISABLED): Disable Modem Status Interrupt 0x1 (ENABLED): Enable Modem Status Interrupt This register is activated when lcr bit7 is 0.	0
ELSI	[2]	R/W	Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. Values: 0x0 (DISABLED): Disable Receiver Line Status Interrupt 0x1 (ENABLED): Enable Receiver Line Status Interrupt This register is activated when lcr bit7 is 0.	0
ETBEI	[1]	R/W	Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. Values: 0x0 (DISABLED): Disable Transmit empty interrupt 0x1 (ENABLED): Enable Transmit empty interrupt This register is activated when lcr bit7 is 0.	0
ERBFI	[0]	R/W	Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFO's enabled). These are the second highest priority interrupts. Values: 0x0 (DISABLED): Disable Receive data Interrupt 0x1 (ENABLED): Enable Receive data Interrupt This register is activated when lcr bit7 is 0.	0

#### 20.4.6 divisor latch high

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x04, 0x04, 0x04, 0x04, 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
DLH	[7:0]	R/W	<p>Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (<code>CLOCK_MODE == Enabled</code>)) frequency divided by sixteen times the value of the baud rate divisor, as follows: <math>\text{baud rate} = (\text{serial clock freq}) / (16 * \text{divisor})</math>.</p> <p>Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLH is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.</p> <p>This register is activated when lcr bit7 is 1.</p>	0

#### 20.4.7 interrupt identity register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x08, 0x08, 0x08, 0x08, 0x08 Reset Value = 0x0000\_0001

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
FIFOSE	[7:6]	R	FIFOs Enabled (or FIFOSE). This is used to indicate whether the FIFOs are enabled or disabled. Values: 0x0 (DISABLED): FIFOs are disabled 0x3 (ENABLED): FIFOs are enabled	0
RSVD	[5:4]	-	Reserved	-
IID	[3:0]	R	Interrupt ID (or IID). This indicates the highest priority pending interrupt which can be one of the following types specified in Values. For information on several levels into which the interrupt priorities are split into, see the 'Interrupts' section in the DW_apb_uart Databook. Note: an interrupt of type 0111 (busy detect) will never get indicated if UART_16550_COMPATIBLE == YES in coreConsultant. Bit 3 indicates an interrupt can only occur when the FIFOs are enabled and used to distinguish a Character Timeout condition interrupt. Values: 0x0 (MODEM_STATUS): modem status 0x1 (NO_INTERRUPT_PENDING): no interrupt pending 0x2 (THR_EMPTY): THR empty 0x4 (RECEIVED_DATA_AVAILABLE): received data available 0x6 (RECEIVER_LINE_STATUS): receiver line status 0x7 (BUSY_DETECT): busy detect 0xc (CHARACTER_TIMEOUT): character timeout	1

#### 20.4.8 FIFO control register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x08, 0x08, 0x08, 0x08, 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-

RT	[7:6]	W	<p>RCVR Trigger (or RT).</p> <p>This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. In auto flow control mode, it is used to determine when the rts_n signal will be de-asserted only when RTC_FCT is disabled. It also determines when the dma_rx_req_n signal will be asserted when in certain modes of operation. For details on DMA support, refer to 'DMA Support' section of data book.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (FIFO_CHAR_1): 1 character in FIFO</li> <li>0x1 (FIFO_QUARTER_FULL): FIFO 1/4 full</li> <li>0x2 (FIFO_HALF_FULL): FIFO 1/2 full</li> <li>0x3 (FIFO_FULL_2): FIFO 2 less than full</li> </ul>	0
TET	[5:4]	W	<p>TX Empty Trigger (or TET).</p> <p>Writes will have no effect when THRE_MODE_USER == Disabled. This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active. It also determines when the dma_tx_req_n signal will be asserted when in certain modes of operation. For details on DMA support, refer to 'DMA Support' section of data book.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (FIFO_EMPTY): FIFO Empty</li> <li>0x1 (FIFO_CHAR_2): 2 characters in FIFO</li> <li>0x2 (FIFO_QUARTER_FULL): FIFO 1/4 full</li> <li>0x3 (FIFO_HALF_FULL): FIFO 1/2 full</li> </ul>	0
DMAM	[3]	W	<p>DMA Mode (or DMAM).</p> <p>This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected (DMA_EXTRA == NO). For details on DMA support, refer to 'DMA Support' section of data book.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (MODE0): Mode 0</li> <li>0x1 (MODE1): Mode 1</li> </ul>	0
XFIFOR	[2]	W	<p>XMIT FIFO Reset (or XFIFOR).</p> <p>This resets the control portion of the transmit FIFO and treats the FIFO as empty. This will also de-assert the DMA TX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'selfclearing' and it is not necessary to clear this bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x1 (RESET): Transmit FIFO reset</li> </ul>	0
RFIFOR	[1]	W	<p>RCVR FIFO Reset (or RFIFOR).</p> <p>This resets the control portion of the receive FIFO and treats the FIFO as empty. This will also de-assert the DMA RX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'selfclearing' and it is not necessary to clear this bit.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x1 (RESET): Receive FIFO reset</li> </ul>	0
FIFOE	[0]	W	<p>FIFO Enable (or FIFOE).</p> <p>This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): FIFO disabled</li> <li>0x1 (ENABLED): FIFO enabled</li> </ul>	0

#### 20.4.9 line control register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])

- Address = Base Address + 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
DLAB	[7]	R/W	<p>Divisor Latch Access Bit.</p> <p>If <code>UART_16550_COMPATIBLE == NO</code> then, writeable only when UART is not busy (<code>USR[0]</code> is zero), otherwise always writable and always readable. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH/LPDLL and LPDLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): Divisor Latch register is writable only when UART Not BUSY</li> <li>0x1 (ENABLED): Divisor Latch register is always readable and writable</li> </ul>	0
BC	[6]	R/W	<p>Break Control Bit.</p> <p>This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by <code>MCR[4]</code>, the <code>sout</code> line is forced low until the Break bit is cleared. If <code>SIR_MODE == Enabled and active</code> (<code>MCR[6]</code> set to one) the <code>sir_out_n</code> line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the <code>sir_out_n</code> line is forced low.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): Serial output is released for data transmission</li> <li>0x1 (ENABLED): Serial output is forced to spacing state</li> </ul>	0
SP	[5]	R/W	<p>Stick Parity.</p> <p>If <code>UART_16550_COMPATIBLE = NO</code>, then writeable only when UART is not busy (<code>USR[0]</code> is 0); otherwise always writable and always readable. This bit is used to force parity value. When <code>PEN</code>, <code>EPS</code> and Stick Parity are set to 1, the parity bit is transmitted and checked as logic 0. If <code>PEN</code> and Stick Parity are set to 1 and <code>EPS</code> is a logic 0, then parity bit is transmitted and checked as a logic 1. If this bit is set to 0, Stick Parity is disabled.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): Stick parity disabled</li> <li>0x1 (ENABLED): Stick parity enabled</li> </ul>	0
EPS	[4]	R/W	<p>Even Parity Select.</p> <p>If <code>UART_16550_COMPATIBLE == NO</code> then, writeable only when UART is not busy (<code>USR[0]</code> is zero), otherwise always writable and always readable. This is used to select between even and odd parity, when parity is enabled (<code>PEN</code> set to one). If set to one, an even number of logic '1's is transmitted or checked. If set to zero, an odd number of logic '1's is transmitted or checked.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (ODD_PARITY): an odd parity is transmitted or checked</li> <li>0x1 (EVEN_PARITY): an even parity is transmitted or checked</li> </ul>	0
PEN	[3]	R/W	<p>Parity Enable.</p> <p>If <code>UART_16550_COMPATIBLE == NO</code> then, writeable only when UART is not busy (<code>USR[0]</code> is zero), otherwise always writable and always readable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): disable parity</li> <li>0x1 (ENABLED): enable parity</li> </ul>	0

STOP	[2]	R/W	<p>Number of stop bits. If <code>UART_16550_COMPATIBLE == NO</code> then, writeable only when UART is not busy (<code>USR[0]</code> is zero), otherwise always writable and always readable. This is used to select the number of stop bits per character that the peripheral will transmit and receive. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (<code>LCR[1:0]</code> set to zero) one and a half stop bits are transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected the receiver will only check the first stop bit.</p> <p>Note: NOTE: The STOP bit duration implemented by <code>DW_apb_uart</code> may appear longer due to idle time inserted between characters for some configurations and baud clock divisor values in the transmit direction; for details on idle time between transmitted transfers, refer to 'Back-to-Back Character Stream Transmission' section in data book.</p> <p>Values: 0x0 (STOP_1BIT): 1 stop bit 0x1 (STOP_1_5BIT_OR_2BIT): 1.5 stop bits when <code>DLS (LCR[1:0])</code> is zero, else 2 stop bit</p>	0
DLS	[1:0]	R/W	<p>Data Length Select (or CLS as used in legacy). If <code>UART_16550_COMPATIBLE == NO</code> then, writeable only when UART is not busy (<code>USR[0]</code> is zero), otherwise always writable and always readable. When <code>DLS_E</code> in <code>LCR_EXT</code> is set to 0, this register is used to select the number of data bits per character that the peripheral will transmit and receive.</p> <p>Values: 0x0 (CHAR_5BITS): 5 data bits per character 0x1 (CHAR_6BITS): 6 data bits per character 0x2 (CHAR_7BITS): 7 data bits per character 0x3 (CHAR_8BITS): 8 data bits per character</p>	0

#### 20.4.10 modem control register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x10, 0x10, 0x10, 0x10, 0x10, 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:7]	-	Reserved	-
SIRE	[6]	R/W	<p>SIR Mode Enable. Writeable only when <code>SIR_MODE == Enabled</code>, always readable. This is used to enable/ disable the IrDA SIR Mode features as described in section 'IrDA 1.0 SIR Protocol' in the databook.</p> <p>Note: To enable SIR mode, write the appropriate value to the MCR register before writing to the LCR register. For details of the recommended programming sequence, refer to 'Programing Examples' section of data book.</p> <p>Values: 0x0 (DISABLED): IrDA SIR Mode disabled 0x1 (ENABLED): IrDA SIR Mode enabled</p>	0
AFCE	[5]	R/W	<p>Auto Flow Control Enable. Writeable only when <code>AFCE_MODE == Enabled</code>, always readable. When FIFOs are enabled and the Auto Flow Control Enable (AFCE) bit is set, Auto Flow Control features are enabled as described in section 'Auto Flow Control' in data book.</p> <p>Values: 0x0 (DISABLED): Auto Flow Control Mode disabled 0x1 (ENABLED): Auto Flow Control Mode enabled</p>	0

LoopBack	[4]	R/W	<p>LoopBack Bit.</p> <p>This is used to put the UART into a diagnostic mode for test purposes. If operating in UART mode (SIR_MODE != Enabled OR NOT active, MCR[6] set to zero), data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally. If operating in infrared mode (SIR_MODE == Enabled AND active, MCR[6] set to one), data on the sir_out_n line is held low, while serial data output is inverted and looped back to the sir_in line.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): Loopback mode disabled</li> <li>0x1 (ENABLED): Loopback mode enabled</li> </ul>	0
OUT2	[3]	R/W	<p>OUT2.</p> <p>This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n. Note that in Loopback mode (MCR[4] set to one), the out2_n output is held inactive high while the value of this location is internally looped back to an input.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (OUT2_0): out2_n de-asserted (logic 1)</li> <li>0x1 (OUT2_1): out2_n asserted (logic 0)</li> </ul>	0
OUT1	[2]	R/W	<p>OUT1.</p> <p>This is used to directly control the user-designated Output1 (out1_n) output. The value written to this location is inverted and driven out on out1_n. Note that in Loopback mode (MCR[4] set to one), the out1_n output is held inactive high while the value of this location is internally looped back to an input.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (OUT1_0): out1_n de-asserted (logic 1)</li> <li>0x1 (OUT1_1): out1_n asserted (logic 0)</li> </ul>	0
RTS	[1]	R/W	<p>Request to Send.</p> <p>This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] set to zero), the rts_n signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, AFCE_MODE == Enabled and active (MCR[5] set to one) and FIFO's enable (FCR[0] set to one), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). The rts_n signal will be de-asserted when MCR[1] is set low.</p> <p>Note that in Loopback mode (MCR[4] set to one), the rts_n output is held inactive high while the value of this location is internally looped back to an input.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (INACTIVE): Request to Send rts_n de-asserted (logic 1)</li> <li>0x1 (ACTIVE): Request to Send rts_n asserted (logic 0)</li> </ul>	0
DTR	[0]	R/W	<p>Data Terminal Ready.</p> <p>This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n.</p> <p>The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications. Note that in Loopback mode (MCR[4] set to one), the dtr_n output is held inactive high while the value of this location is internally looped back to an input.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (INACTIVE): dtr_n de-asserted (logic1)</li> <li>0x1 (ACTIVE): dtr_n asserted (logic 0)</li> </ul>	0

### 20.4.11 line status register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x14, 0x14, 0x14, 0x14, 0x14 Reset Value = 0x0000\_0060

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
ADDR_RCVD	[8]	R	<p>Address Received Bit.            If 9Bit data mode (LCR_EXT[0]=1) is enabled, this bit is used to indicate the 9th bit of the receive data is set to 1.            This bit can also be used to indicate whether the incoming character is address or data.            1 = Indicates the character is address.            0 = Indicates the character is data.</p> <p>In the FIFO mode, since the 9th bit is associated with a character received, it is revealed when the character with the 9th bit set to 1 is at the top of the FIFO.</p> <p>Reading the LSR clears the 9BIT.</p> <p>Note: User needs to ensure that interrupt gets cleared (reading LSR register) before the next address byte arrives. If there is a delay in clearing the interrupt, then Software will not be able to distinguish between multiple address related interrupt.</p>	0
RFE	[7]	R	<p>Receiver FIFO Error bit.            This bit is only relevant when FIFO_MODE != NONE AND FIFO's are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO.</p> <p>This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.</p> <p>Values:            0x0 (NO_RX_FIFO_ERROR): No error in RX FIFO            0x1 (RX_FIFO_ERROR): Error in RX FIFO</p>	0
TEM	[6]	R	<p>Transmitter Empty bit.            If in FIFO mode (FIFO_MODE != NONE) and FIFO's enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If in the non-FIFO mode or FIFO's are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.</p> <p>Values:            0x0 (DISABLED): Transmitter not empty            0x1 (ENABLED): Transmitter empty</p>	1
THRE	[5]	R	<p>Transmit Holding Register Empty bit.            If THRE_MODE_USER = Disabled or THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty.</p> <p>This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If THRE_MODE_USER == Enabled AND FIFO_MODE != NONE and both modes are active (IER[7] set to one and FCR[0] set to one respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting. Programmable THRE interrupt mode operation is described in detail in section 'Programmable THRE Interrupt' in data book.</p> <p>Values:            0x0 (DISABLED): THRE interrupt control is disabled            0x1 (ENABLED): THRE interrupt control is enabled</p>	1

BI	[4]	R	<p>Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. If in UART mode it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits. If in infrared mode it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART. In the FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit (if LSR_STATUS_CLEAR==1) Or Reading the LSR or RBR clears the BI bit (if LSR_STATUS_CLEAR==0). In the non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read. Note: If a FIFO is full when a break condition is received, a FIFO overrun occurs. The break condition and all the information associated with it-parity and framing errors-is discarded; any information that a break character was received is lost. Values: 0x0 (NO_BREAK): No break sequence detected 0x1 (BREAK): Break sequence detected</p>	0
FE	[3]	R	<p>Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs the UART will try resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit i.e. data, and/or parity and stop. It should be noted that the Framing Error (FE) bit (LSR[3]) will be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). This happens because the break character implicitly generates a framing error by holding the sin input to logic 0 for longer than the duration of a character. Reading the LSR clears the FE bit (if LSR_STATUS_CLEAR==1) Or Reading the LSR or RBR clears the FE bit (if LSR_STATUS_CLEAR==0). Values: 0x0 (NO_FRAMING_ERROR): no framing error 0x1 (FRAMING_ERROR): framing error</p>	0
PE	[2]	R	<p>Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO. It should be noted that the Parity Error (PE) bit (LSR[2]) will be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). In this situation, the Parity Error bit is set if parity generation and detection is enabled (LCR[3]==1) and the parity is set to odd (LCR[4]==0). Reading the LSR clears the PE bit (if LSR_STATUS_CLEAR==1) Or Reading the LSR or RBR clears the PE bit (if LSR_STATUS_CLEAR==0). Values: 0x0 (NO_PARITY_ERROR): no parity error 0x1 (PARITY_ERROR): parity error</p>	0

OE	[1]	R	<p>Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read. In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost. Reading the LSR clears the OE bit (if LSR_STATUS_CLEAR==1) Or Reading the LSR or RBR clears the OE bit (if LSR_STATUS_CLEAR==0). Values: 0x0 (NO_OVER_RUN_ERROR): no overrun error 0x1 (OVER_RUN_ERROR): overrun error</p>	0
DR	[0]	R	<p>Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. This bit is cleared when the RBR is read in the non-FIFO mode, or when the receiver FIFO is empty, in the FIFO mode. Values: 0x0 (NOT_READY): data not ready 0x1 (READY): data ready</p>	0

#### 20.4.12 modem status register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x18, 0x18, 0x18, 0x18, 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
DCD	[7]	R	<p>Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n. That is this bit is the complement dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set. In Loopback Mode (MCR[4] set to one), DCD is the same as MCR[3] (Out2). Values: 0x0 (DEASSERTED): dcd_n input is de-asserted (logic 1) 0x1 (ASSERTED): dcd_n input is asserted (logic 0)</p>	0
RI	[6]	R	<p>Ring Indicator. This is used to indicate the current state of the modem control line ri_n. That is this bit is the complement ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set. In Loopback Mode (MCR[4] set to one), RI is the same as MCR[2] (Out1). Values: 0x0 (DEASSERTED): ri_n input is de-asserted (logic 1) 0x1 (ASSERTED): ri_n input is asserted (logic 0)</p>	0

DSR	[5]	R	<p>Data Set Ready. This is used to indicate the current state of the modem control line <code>dsr_n</code>. That is this bit is the complement <code>dsr_n</code>. When the Data Set Ready input (<code>dsr_n</code>) is asserted it is an indication that the modem or data set is ready to establish communications with the DW_apb_uart.</p> <p>In Loopback Mode (MCR[4] set to one), DSR is the same as MCR[0] (DTR).</p> <p>Values: 0x0 (DEASSERTED): <code>dsr_n</code> input is de-asserted (logic 1) 0x1 (ASSERTED): <code>dsr_n</code> input is asserted (logic 0)</p>	0
CTS	[4]	R	<p>Clear to Send. This is used to indicate the current state of the modem control line <code>cts_n</code>. That is, this bit is the complement <code>cts_n</code>. When the Clear to Send input (<code>cts_n</code>) is asserted it is an indication that the modem or data set is ready to exchange data with the DW_apb_uart.</p> <p>In Loopback Mode (MCR[4] set to one), CTS is the same as MCR[1] (RTS).</p> <p>Values: 0x0 (DEASSERTED): <code>cts_n</code> input is de-asserted (logic 1) 0x1 (ASSERTED): <code>cts_n</code> input is asserted (logic 0)</p>	0
DDCD	[3]	R	<p>Delta Data Carrier Detect. This is used to indicate that the modem control line <code>dcd_n</code> has changed since the last time the MSR was read. Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] set to one), DDCD reflects changes on MCR[3] (Out2).</p> <p>Note, if the DDCD bit is not set and the <code>dcd_n</code> signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit will get set when the reset is removed if the <code>dcd_n</code> signal remains asserted.</p> <p>Values: 0x0 (NO_CHANGE): No change on <code>dcd_n</code> since last read of MSR 0x1 (CHANGE): change on <code>dcd_n</code> since last read of MSR</p>	0
TERI	[2]	R	<p>Trailing Edge of Ring Indicator. This is used to indicate that a change on the input <code>ri_n</code> (from an active low, to an inactive high state) has occurred since the last time the MSR was read.</p> <p>Reading the MSR clears the TERI bit. In Loopback Mode (MCR[4] set to one), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.</p> <p>Values: 0x0 (NO_CHANGE): no change on <code>ri_n</code> since last read of MSR 0x1 (CHANGE): change on <code>ri_n</code> since last read of MSR</p>	0
DDSR	[1]	R	<p>Delta Data Set Ready. This is used to indicate that the modem control line <code>dsr_n</code> has changed since the last time the MSR was read.</p> <p>Reading the MSR clears the DDSR bit. In Loopback Mode (MCR[4] set to one), DDSR reflects changes on MCR[0] (DTR).</p> <p>Note, if the DDSR bit is not set and the <code>dsr_n</code> signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit will get set when the reset is removed if the <code>dsr_n</code> signal remains asserted.</p> <p>Values: 0x0 (NO_CHANGE): no change on <code>dsr_n</code> since last read of MSR 0x1 (CHANGE): change on <code>dsr_n</code> since last read of MSR</p>	0
DCTS	[0]	R	<p>Delta Clear to Send. This is used to indicate that the modem control line <code>cts_n</code> has changed since the last time the MSR was read.</p> <p>Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] set to one), DCTS reflects changes on MCR[1] (RTS).</p> <p>Note, if the DCTS bit is not set and the <code>cts_n</code> signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit will get set when the reset is removed if the <code>cts_n</code> signal remains asserted.</p> <p>Values: 0x0 (NO_CHANGE): no change on <code>cts_n</code> since last read of MSR 0x1 (CHANGE): change on <code>cts_n</code> since last read of MSR</p>	0

### 20.4.13 scratch register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x1C, 0x1C, 0x1C, 0x1C, 0x1C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
SCR	[7:0]	R/W	This register is for programmers to use as a temporary storage space. It has no defined purpose in the DW_apb_uart.	0

### 20.4.14 Low Power Divisor Latch Low

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x20, 0x20, 0x20, 0x20, 0x20, 0x20 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
LPDLL	[7:0]	R/W	<p>This register makes up the lower 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver.</p> <p>The output low power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> <p>Low power baud rate = (serial clock freq) / (16 * divisor) Therefore a divisor must be selected to give a baud rate of 115.2K.</p> <p>Note: When the Low Power Divisor Latch Registers (LPDLL and LPDLH) are set to zero, the low power baud clock is disabled and no low power pulse detection (or any pulse detection for that matter) will occur at the receiver. Also, once the LPDLL is set at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.</p>	0

### 20.4.15 Low Power Divisor Latch Low

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x24, 0x24, 0x24, 0x24, 0x24 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
LPDLH	[7:0]	R/W	<p>This register makes up the upper 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver.</p> <p>The output low power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> <p>Low power baud rate = (serial clock freq) / (16 * divisor) Therefore a divisor must be selected to give a baud rate of 115.2K.</p> <p>Note: When the Low Power Divisor Latch Registers (LPDLL and LPDLH) are set to zero, the low power baud clock is disabled and no low power pulse detection (or any pulse detection for that matter) will occur at the receiver. Also, once the LPDLH is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.</p>	0

### 20.4.16 shadow receive buffer registers

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x30, 0x30, 0x30, 0x30, 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
SRBRn	[8:0]	R	<p>Shadow Receive Buffer Register n. This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>This register is activated when lcr [7] is 0.</p>	0

### 20.4.17 shadow transmit holding registers

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x30, 0x30, 0x30, 0x30, 0x30 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
STHRn	[8:0]	W	<p>Shadow Transmit Holding Register n. This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.</p> <p>If in non-FIFO mode or FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.</p> <p>If in FIFO mode and FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> <p>This register is activated when lcr [7] is 0.</p>	0

### 20.4.18 FIFO access register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x70, 0x70, 0x70, 0x70, 0x70, 0x70 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
FAR	[0]	R/W	<p>Writes will have no effect when FIFO_ACCESS == No, always readable. This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFO's are implemented and enabled.</p> <p>When FIFOs are not implemented or not enabled it allows the RBR to be written by the master and the THR to be read by the master. Note, that when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFO's are treated as empty.</p> <p>Values:</p> <p>0x0 (DISABLED): FIFO access mode disabled 0x1 (ENABLED): FIFO access mode enabled</p>	0

### 20.4.19 transmit FIFO read

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x74, 0x74, 0x74, 0x74, 0x74 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
TFR	[7:0]	R	<p>Transmit FIFO Read. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are implemented and enabled, reading this register gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO. When FIFO's are not implemented or not enabled, reading this register gives the data in the THR.</p>	0

### 20.4.20 receiver FIFO write

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x78, 0x78, 0x78, 0x78, 0x78 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:10]	-	Reserved	-
RFFE	[9]	W	<p>Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are implemented and enabled, this bit is used to write framing error detection information to the receive FIFO. When FIFO's are not implemented or not enabled, this bit is used to write framing error detection information to the RBR.</p>	0
RFPE	[8]	W	<p>Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are implemented and enabled, this bit is used to write parity error detection information to the receive FIFO. When FIFO's are not implemented or not enabled, this bit is used to write parity error detection information to the RBR. Values: 0x0 (DISABLED): Parity error disabled 0x1 (ENABLED): Parity error enabled</p>	0

RFWD	[7:0]	W	Receive FIFO Write Data. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are implemented and enabled, the data that is written to the RFWD is pushed into the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFO's are not implemented or not enabled, the data that is written to the RFWD is pushed into the RBR.	0
------	-------	---	---	---

### 20.4.21 uart status register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x7C, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C Reset Value = 0x0000\_0006

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	Reserved	-
RFF	[4]	R	<p>Receive FIFO Full. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the receive FIFO is completely full. That is: This bit is cleared when the RX FIFO is no longer full. Values: 0x0 (NOT_FULL): Receive FIFO not full 0x1 (FULL): Receive FIFO full</p>	0
RFNE	[3]	R	<p>Receive FIFO Not Empty. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the receive FIFO contains one or more entries. This bit is cleared when the RX FIFO is empty. Values: 0x0 (EMPTY): Receive FIFO is empty 0x1 (NOT_EMPTY): Receive FIFO is not empty</p>	0
TFE	[2]	R	<p>Transmit FIFO Empty. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the transmit FIFO is completely empty. This bit is cleared when the TX FIFO is no longer empty. Values: 0x0 (NOT_EMPTY): Transmit FIFO is not empty 0x1 (EMPTY): Transmit FIFO is empty</p>	1
TFNF	[1]	R	<p>Transmit FIFO Not Full. This bit is only valid when FIFO_STAT == YES. This is used to indicate that the transmit FIFO is not full. This bit is cleared when the TX FIFO is full. Values: 0x0 (FULL): Transmit FIFO is full 0x1 (NOT_FULL): Transmit FIFO is not full</p>	1
BUSY	[0]	R	<p>UART Busy. This bit is only valid when UART_16550_COMPATIBLE == NO. This indicates that a serial transfer is in progress, when cleared indicates that the DW_apb_uart is idle or inactive. This bit will be set to 1 (busy) under any of the following conditions:            - Transmission in progress on serial interface            - Transmit data present in THR, when FIFO access mode is not being used (FAR = 0) and the baud divisor is non-zero (DLH,DLL does not equal 0) when the divisor latch access bit is 0 (LCR.DLAB = 0)            - Reception in progress on the interface            - Receive data present in RBR, when FIFO access mode is not being used (FAR = 0)            Note: It is possible for the UART Busy bit to be cleared even though a new character may have been sent from another device. That is, if the DW_apb_uart has no data in the THR and RBR and there is no transmission in progress and a start bit of a new character has just reached the DW_apb_uart. This is due to the fact that a valid start is not seen until the middle of the bit period and this duration is dependent on the baud divisor that has been programmed. If a second system clock has been implemented (CLOCK_MODE == Enabled), the assertion of this bit will also be delayed by several cycles of the slower clock.            Values:            0x0 (IDLE): DW_apb_uart is idle or inactive            0x1 (BUSY): DW_apb_uart is busy (actively transferring data)</p>	0

#### 20.4.22 transmit FIFO level

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x80, 0x80, 0x80, 0x80, 0x80 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
TFL	[8:0]	R	Transmit FIFO Level. This indicates the number of data entries in the transmit FIFO.	0

#### 20.4.23 receive FIFO level

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x84, 0x84, 0x84, 0x84, 0x84 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:9]	-	Reserved	-
RFL	[8:0]	R	Receive FIFO Level. This indicates the number of data entries in the receive FIFO.	0

#### 20.4.24 software reset register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x88, 0x88, 0x88, 0x88, 0x88 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
XFR	[2]	W	s XMIT FIFO Reset. Writes will have no effect when FIFO_MODE == NONE. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This will also de-assert the DMA TX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES). Note that	0
RFR	[1]	W	RCVR FIFO Reset. Writes will have no effect when FIFO_MODE == NONE. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty. This will also de-assert the DMA RX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA == YES). Note that this bit is 'selfclearing' and it is not necessary to clear this bit.	0
UR	[0]	W	UART Reset. This asynchronously resets the DW_apb_uart and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains will be reset.	0

#### 20.4.25 shadow request to send

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x8C, 0x8C, 0x8C, 0x8C, 0x8C, 0x8C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SRTS	[0]	R/W	Shadow Request to Send. This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to perform a read modify write on the MCR. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] set to zero), the rts_n signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, AFCE_MODE == Enabled and active (MCR[5] set to one) and FIFO's enable (FCR[0] set to one), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold) only when RTC Flow Trigger is disabled; otherwise it is gated by the receiver FIFO almost-full trigger, where 'almost full' refers to two available slots in the FIFO (rts_n is inactive high when above the threshold). Note that in Loopback mode (MCR[4] set to one), the rts_n output is held inactive high while the value of this location is internally looped back to an input. Values: 0x0 (DEASSERTED): Shadow Request to Send uart_rts_n logic1 0x1 (ASSERTED): Shadow Request to Send uart_rts_n logic0	0

#### 20.4.26 shadow request to send

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x90, 0x90, 0x90, 0x90, 0x90, 0x90 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SBCB	[0]	R/W	<p>Shadow Break Control Bit.            This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to perform a read modify write on the LCR. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared.            If SIR_MODE == Enabled and active (MCR[6] set to one) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver.            Values:            0x0 (NO_BREAK): No spacing on serial output            0x1 (BREAK): Serial output forced to the spacing</p>	0

#### 20.4.27 shadow break control

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x94, 0x94, 0x94, 0x94, 0x94, 0x94 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SDMAM	[0]	R/W	<p>Shadow DMA Mode.            This is a shadow register for the DMA mode bit (FCR[3]).            This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the DMA Mode bit gets updated.            This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected (DMA_EXTRA == NO). See section 5.9 on page 54 for details on DMA support.            Values:            0x0 (MODE_0): Mode 0            0x1 (MODE_1): Mode 1</p>	0

### 20.4.28 shadow FIFO enable

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x98, 0x98, 0x98, 0x98, 0x98, 0x98 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
SFE	[0]	R/W	<p>Shadow FIFO Enable.            This is a shadow register for the FIFO enable bit (FCR[0]).            This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated. This enables/disables the transmit (XMIT) and receive (RCVR) FIFO's. If this bit is set to zero (disabled) after being enabled then both the XMIT and RCVR controller portion of FIFO's will be reset.</p> <p>Values:            0x0 (DISABLED): FIFOs are disabled            0x1 (ENABLED): FIFOs are enabled</p>	0

### 20.4.29 shadow receiver trigger

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0x9C, 0x9C, 0x9C, 0x9C, 0x9C, 0x9C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
SRT	[1:0]	R/W	<p>Shadow RCVR Trigger.            This is a shadow register for the RCVR trigger bits (FCR[7:6]).            This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the RCVR trigger bit gets updated.</p> <p>This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. It also determines when the dma_rx_req_n signal will be asserted when DMA Mode (FCR[3]) is set to one.</p> <p>Values:            0x0 (FIFO_CHAR_1): 1 character in FIFO            0x1 (FIFO_QUARTER_FULL): FIFO 1/4 full            0x2 (FIFO_HALF_FULL): FIFO 1/2 full            0x3 (FIFO_FULL_2): FIFO 2 less than full</p>	0

### 20.4.30 shadow transmitter trigger

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:2]	-	Reserved	-
STET	[1:0]	R/W	<p>Shadow TX Empty Trigger.            This is a shadow register for the TX empty trigger bits (FCR[5:4]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the TX empty trigger bit gets updated. Writes will have no effect when THRE_MODE_USER == Disabled. This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (FIFO_EMPTY): FIFO empty</li> <li>0x1 (FIFO_CHAR_2): 2 characters in FIFO</li> <li>0x2 (FIFO_QUARTER_FULL): FIFO 1/4 full</li> <li>0x3 (FIFO_HALF_FULL): FIFO 1/2 full</li> </ul>	0

### 20.4.31 halt Tx

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xA4, 0xA4, 0xA4, 0xA4, 0xA4, 0xA4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
HTX	[0]	R	<p>Halt TX.            Writes will have no effect when FIFO_MODE == NONE, always readable. This register is used to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFO's are implemented and enabled.</p> <p>Note, if FIFO's are implemented and not enabled the setting of the halt TX register will have no effect on operation.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>0x0 (DISABLED): Halt Transmission disabled</li> <li>0x1 (ENABLED): Halt Transmission enabled</li> </ul>	0

### 20.4.32 dma software acknowledge

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xA8, 0xA8, 0xA8, 0xA8, 0xA8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DMASA	[0]	W	<p>DMA Software Acknowledge.            Writes will have no effect when DMA_EXTRA == No. This register is use to perform DMA software acknowledge if a transfer needs to be terminated due to an error condition. For example, if the DMA disables the channel, then the DW_apb_uart should clear its request. This will cause the TX request, TX single, RX request and RX single signals to deassert.            Note that this bit is 'self-clearing' and it is not necessary to clear this bit.            Values:            0x1 (SOFT_ACK): DMA software acknowledge</p>	0

### 20.4.33 Transceiver Control Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xAC, 0xAC, 0xAC, 0xAC, 0xAC, 0xAC Reset Value = 0x0000\_0006

Name	Bit	Type	Description	Reset Value
RSVD	[31:5]	-	Reserved	-
XFER_MODE	[4:3]	R/W	<p>Transfer Mode.</p> <p>0: In this mode, transmit and receive can happen simultaneously. The user can enable DE_EN, RE_EN at any point of time. Turn around timing as programmed in the TAT register is not applicable in this mode.</p> <p>1: In this mode, DE and RE are mutually exclusive. Either DE or RE only one of them is expected to be enabled through programming.</p> <p>Hardware will consider the Turn Around timings which are programmed in the TAT register while switching from RE to DE or DE to RE. For transmission Hardware will wait if it is in middle of receiving any transfer, before it starts transmitting.</p> <p>2: In this mode, DE and RE are mutually exclusive. Once DE_EN/RE_EN is programmed - by default 're' will be enabled and DW_apb_uart controller will be ready to receive. If the user programs the TX FIFO with the data then DW_apb_uart, after ensuring no receive is in progress, disable 're' and enable 'de' signal.</p> <p>Once the TX FIFO becomes empty, 're' signal gets enabled and 'de' signal will be disabled. In this mode of operation hardware will consider the Turn Around timings which are programmed in the TAT register while switching from RE to DE or DE to RE. In this mode, 'de' and 're' signals are strictly complementary to each other.</p>	0
DE_POL	[2]	R/W	<p>Driver Enable Polarity.</p> <p>1: DE signal is active high</p> <p>0: DE signal is active low</p>	1
RE_POL	[1]	R/W	<p>Receiver Enable Polarity.</p> <p>1: RE signal is active high</p> <p>0: RE signal is active low</p> <p>Reset Value: UART_RE_POL</p>	1
RS485_EN	[0]	R/W	<p>RS485 Transfer Enable.</p> <p>0 : In this mode, the transfers are still in the RS232 mode.</p> <p>All other fields in this register are reserved and register DE_EN/RE_EN/TAT are also reserved.</p> <p>1 : In this mode, the transfers will happen in RS485 mode.</p> <p>All other fields of this register are applicable.</p>	0

### 20.4.34 Driver Output Enable Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])

- Address = Base Address + 0xB0, 0xB0, 0xB0, 0xB0, 0xB0, 0xB0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
DE_Enable	[0]	R/W	DE Enable control. The 'DE Enable' register bit is used to control assertion and de-assertion of 'de' signal. - 0: De-assert 'de' signal - 1: Assert 'de' signal	0

#### 20.4.35 Receiver Output Enable Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xB4, 0xB4, 0xB4, 0xB4, 0xB4, 0xB4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	-	Reserved	-
RE_Enable	[0]	R/W	RE Enable control. The 'RE Enable' register bit is used to control assertion and de-assertion of 're' signal. 0: De-assert 're' signal 1: Assert 're' signal	0

#### 20.4.36 Driver Output Enable Timing Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xB8, 0xB8, 0xB8, 0xB8, 0xB8, 0xB8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
DE_De-assertion_Time	[23:16]	R/W	Driver Enable de-assertion time. This field controls the amount of time (in terms of number of serial clock periods) between the end of stop bit on the sout to the falling edge of Driver output enable signal.	0
RSVD	[15:8]	-	Reserved	-

DE_A Assertion_Time	[7:0]	R/W	Driver Enable assertion time. This field controls the amount of time (in terms of number of serial clock periods) between the assertion of rising edge of Driver output enable signal to serial transmit enable. Any data in transmit buffer, will start on serial output (sout) after the transmit enable.	0
---------------------	-------	-----	--	---

#### 20.4.37 TurnAround Timing Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xBC, 0xBC, 0xBC, 0xBC, 0xBC, 0xBC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RE_to_DE	[31:16]	R/W	Receiver Enable to Driver Enable TurnAround time. Turnaround time (in terms of serial clock) for RE Deassertion to DE assertion. Note: - If the DE assertion time in the DET register is 0, then the actual value is the programmed value + 3. - If the DE assertion time in the DET register is 1, then the actual value is the programmed value + 2. - If the DE assertion time in the DET register is greater than 1, then the actual value is the programmed value + 1.	0
DE_to_RE	[15:0]	R/W	Driver Enable to Receiver Enable TurnAround time. Turnaround time (in terms of serial clock) for DE Deassertion to RE assertion. Note: The actual time is the programmed value + 1.	0

#### 20.4.38 Divisor Latch Fraction Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
DLF	[5:0]	R/W	Fractional part of divisor. The fractional value is added to integer value set by DLH, DLL. Fractional value is determined by (Divisor Fraction value)/(2 ^DLF_SIZE). For information on DLF values to be programmed for DLF_SIZE=4, see the 'Fractional Baud Rate Support' section in the DW_apb_uart Databook.	0

### 20.4.39 Receive Address Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
RAR	[7:0]	R/W	<p>This is an address matching register during receive mode. If the 9-th bit is set in the incoming character then the remaining 8-bits will be checked against this register value.</p> <p>If the match happens then sub-sequent characters with 9-th bit set to 0 will be treated as data byte until the next address byte is received.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>- This register is applicable only when 'ADDR_MATCH'(LCR_EXT[1] and 'DLS_E' (LCR_EXT[0]) bits are set to 1.</li> <li>- If UART_16550_COMPATIBLE is configured to 0, then RAR should be programmed only when UART is not busy.</li> <li>- If UART_16550_COMPATIBLE is configured to 0, then RAR can be programmed at any point of the time. However, user must not change this register value when any receive is in progress.</li> </ul>	0

### 20.4.40 Transmit Address Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xC8, 0xC8, 0xC8, 0xC8, 0xC8, 0xC8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	-	Reserved	-
TAR	[7:0]	R/W	<p>This is an address matching register during transmit mode. If DLS_E (LCR_EXT[0]) bit is enabled, then DW_apb_uart will send the 9-bit character with 9-th bit set to 1 and remaining 8-bit address will be sent from this register provided 'SEND_ADDR' (LCR_EXT[2]) bit is set to 1.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>- This register is used only to send the address. The normal data should be sent by programming THR register.</li> <li>- Once the address is started to send on the DW_apb_uart serial lane, then 'SEND_ADDR' bit will be auto-cleared by the hardware.</li> </ul>	0

#### 20.4.41 Line Extended Control Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:4]	-	Reserved	-
TRANSMIT_MODE	[3]	R/W	<p>Transmit mode control bit. This bit is used to control the type of transmit mode during 9-bit data transfers.</p> <p>1: In this mode of operation, Transmit Holding Register (THR) and Shadow Transmit Holding Register (STHR) are 9-bit wide. The user needs to ensure that the THR/ STHR register is written correctly for address/data.</p> <p>Address: 9th bit is set to 1, Data : 9th bit is set to 0. Note: Transmit address register (TAR) is not applicable in this mode of operation.</p> <p>0: In this mode of operation, Transmit Holding Register (THR) and Shadow Transmit Holding register (STHR) are 8-bit wide. The user needs to program the address into Transmit Address Register (TAR) and data into the THR/ STHR register. SEND_ADDR bit is used as a control knob to indicate the DW_apb_uart on when to send the address.</p>	0
SEND_ADDR	[2]	R/W	<p>Send address control bit. This bit is used as a control knob for the user to determine when to send the address during transmit mode.</p> <p>1 = 9-bit character will be transmitted with 9-th bit set to 1 and the remaining 8-bits will match to what is being programmed in 'Transmit Address Register'.</p> <p>0 = 9-bit character will be transmitted with 9-th bit set to 0 and the remaining 8-bits will be taken from the TXFIFO which is programmed through 8-bit wide THR/STHR register.</p> <p>Note:</p> <ol style="list-style-type: none"> <li>1. This bit is auto-cleared by the hardware, after sending out the address character. User is not expected to program this bit to 0.</li> <li>2. This field is applicable only when DLS_E bit is set to 1 and TRANSMIT_MODE is set to 0.</li> </ol>	0
ADDR_MATCH	[1]	R/W	<p>Address Match Mode. This bit is used to enable the address match feature during receive.</p> <p>1 = Address match mode; DW_apb_uart will wait until the incoming character with 9-th bit set to 1. And further checks to see if the address matches with what is programmed in 'Receive Address Match Register'. If match is found, then sub-sequent characters will be treated as valid data and DW_apb_uart starts receiving data.</p> <p>0 = Normal mode; DW_apb_uart will start to receive the data and 9-bit character will be formed and written into the receive RXFIFO. User is responsible to read the data and differentiate b/n address and data.</p> <p>Note: This field is applicable only when DLS_E is set to 1.</p>	0
DLS_E	[0]	R/W	Extension for DLS. This bit is used to enable 9-bit data for transmit and receive transfers.	0

### 20.4.42 Component Parameter Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
FIFO_MODE	[23:16]	R	Encoding of FIFO_MODE configuration parameter value. Values: 0x0 (FIFO_MODE_0): FIFO mode is 0 0x1 (FIFO_MODE_16): FIFO mode is 16 0x2 (FIFO_MODE_32): FIFO mode is 32 0x4 (FIFO_MODE_64): FIFO mode is 64 0x8 (FIFO_MODE_128): FIFO mode is 128 0x10 (FIFO_MODE_256): FIFO mode is 256 0x20 (FIFO_MODE_512): FIFO mode is 512 0x40 (FIFO_MODE_1024): FIFO mode is 1024 0x80 (FIFO_MODE_2048): FIFO mode is 2048	0
RSVD	[15:14]	-	Reserved	-
DMA_EXTRA	[13]	R	Encoding of DMA_EXTRA configuration parameter value. Values: 0x0 (DISABLED): DMA_EXTRA disabled 0x1 (ENABLED): DMA_EXTRA enabled	0
UART_ADD_ENCODED_PARAMS	[12]	R	Encoding of UART_ADD_ENCODED_PARAMS configuration parameter value. Values: 0x0 (DISABLED): UART_ADD_ENCODED_PARAMS disabled 0x1 (ENABLED): UART_ADD_ENCODED_PARAMS enabled	0
SHADOW	[11]	R	Encoding of SHADOW configuration parameter value. Values: 0x0 (DISABLED): SHADOW disabled 0x1 (ENABLED): SHADOW enabled	0
FIFO_STAT	[10]	R	Encoding of FIFO_STAT configuration parameter value. Values: 0x0 (DISABLED): FIFO_STAT disabled 0x1 (ENABLED): FIFO_STAT enabled	0
FIFO_ACCESS	[9]	R	Encoding of FIFO_ACCESS configuration parameter value. Values: 0x0 (DISABLED): FIFO_ACCESS disabled 0x1 (ENABLED): FIFO ACCESS enabled	0
ADDITIONAL_FEAT	[8]	R	Encoding of ADDITIONAL_FEATURES configuration parameter value. Values: 0x0 (DISABLED): Additional features disabled 0x1 (ENABLED): Additional features enabled	0
SIR_LP_MODE	[7]	R	R Encoding of SIR_LP_MODE configuration parameter value. Values: 0x0 (DISABLED): SIR_LP mode disabled 0x1 (ENABLED): SIR_LP mode enabled	0
SIR_MODE	[6]	R	Encoding of SIR_MODE configuration parameter value. Values: 0x0 (DISABLED): SIR mode disabled 0x1 (ENABLED): SIR mode enabled	0
THRE_MODE	[5]	R	Encoding of THRE_MODE configuration parameter value. Values: 0x0 (DISABLED): THRE mode disabled 0x1 (ENABLED): THRE mode enabled	0
AFCE_MODE	[4]	R	Encoding of AFCE_MODE configuration parameter value. Values: 0x0 (DISABLED): AFCE mode disabled 0x1 (ENABLED): AFCE mode enabled	0

RSVD	[3:2]	-	Reserved	-
APB_DATA_WIDTH	[1:0]	R	Encoding of APB_DATA_WIDTH configuration parameter value. Values: 0x0 (APB_8BITS): APB data width is 8 bits 0x1 (APB_16BITS): APB data width is 16 bits 0x2 (APB_32BITS): APB data width is 32 bits	0

#### 20.4.43 UART Component Version

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
UART_Component_Version	[31:0]	R	ASCII value for each number in the version, followed by *. For example 32_30_31_2A represents the version 2.01*	0

#### 20.4.44 Component Type Register

- Base Address: 2088\_0000(UART[0])
- Base Address: 2089\_0000(UART[1])
- Base Address: 208A\_0000(USART[0])
- Base Address: 208B\_0000(USART[1])
- Base Address: 208C\_0000(USART[2])
- Base Address: 208D\_0000(USART[3])
- Address = Base Address + 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC Reset Value = 0x4457\_0110

Name	Bit	Type	Description	Reset Value
Peripheral_ID	[31:0]	R	This register contains the peripherals identification code.	0x44570110

# 21 PWM/TIMER

## 21.1 Overview

The DRONE\_SOC has three 32-bit Pulse Width Modulation (PWM) timers PWM, SECURE\_TIMER, TIMER. These Timers generate internal interrupts for the ARM subsystem. Additionally, Timers 0, 1, 2 and 3 include a PWM function that drives an external I/O signal. The PWM has a dead-zone generator capability to support a large current device.

The Timers use the TCLK 0, 1, 2, 3 and Oscillator Clock as source clock. All Timers have a programmable 8-bit prescaler that provides the first level of division for the TCLK. Each timer has its own private clock-divider that provides a second level of clock division (prescaler divided by 2, 4, 8, or 16).

Each Timer has its own 32-bit down-counter which is driven by the timer clock. The down-counter is initially loaded from the Timer Count Buffer register (TCNTBn). When the down-counter reaches zero, the timer interrupt request is generated to inform the CPU that the Timer operation is complete. When the Timer down-counter reaches zero, the value of corresponding TCNTBn automatically reloads into the down-counter to start the next cycle. However, when the Timer stops, for example, by clearing the timer enable bit of TCONn during the Timer running mode, the value of TCNTBn is not reloaded into the counter.

The PWM function uses the value of the TCMPBn register. The Timer Control Logic changes the output level if down-counter value matches the value of the compare register in Timer Control Logic. Therefore, the compare register determines the turn-on time or turn-off time of a PWM output.

The TCNTBn and TCMPBn registers are double buffered so that it allows the Timer parameters to be updated in the middle of a cycle. The new values do not take effect until the current Timer cycle is completed.

Figure 21.1 illustrates an example of a PWM cycle:

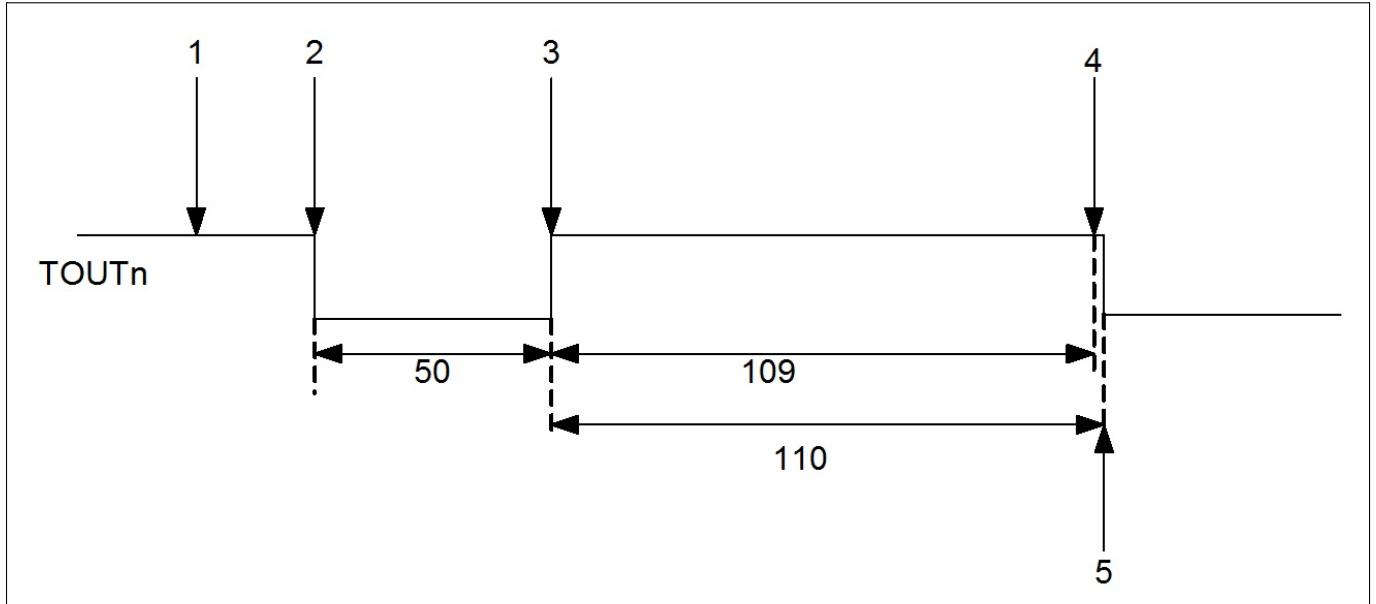
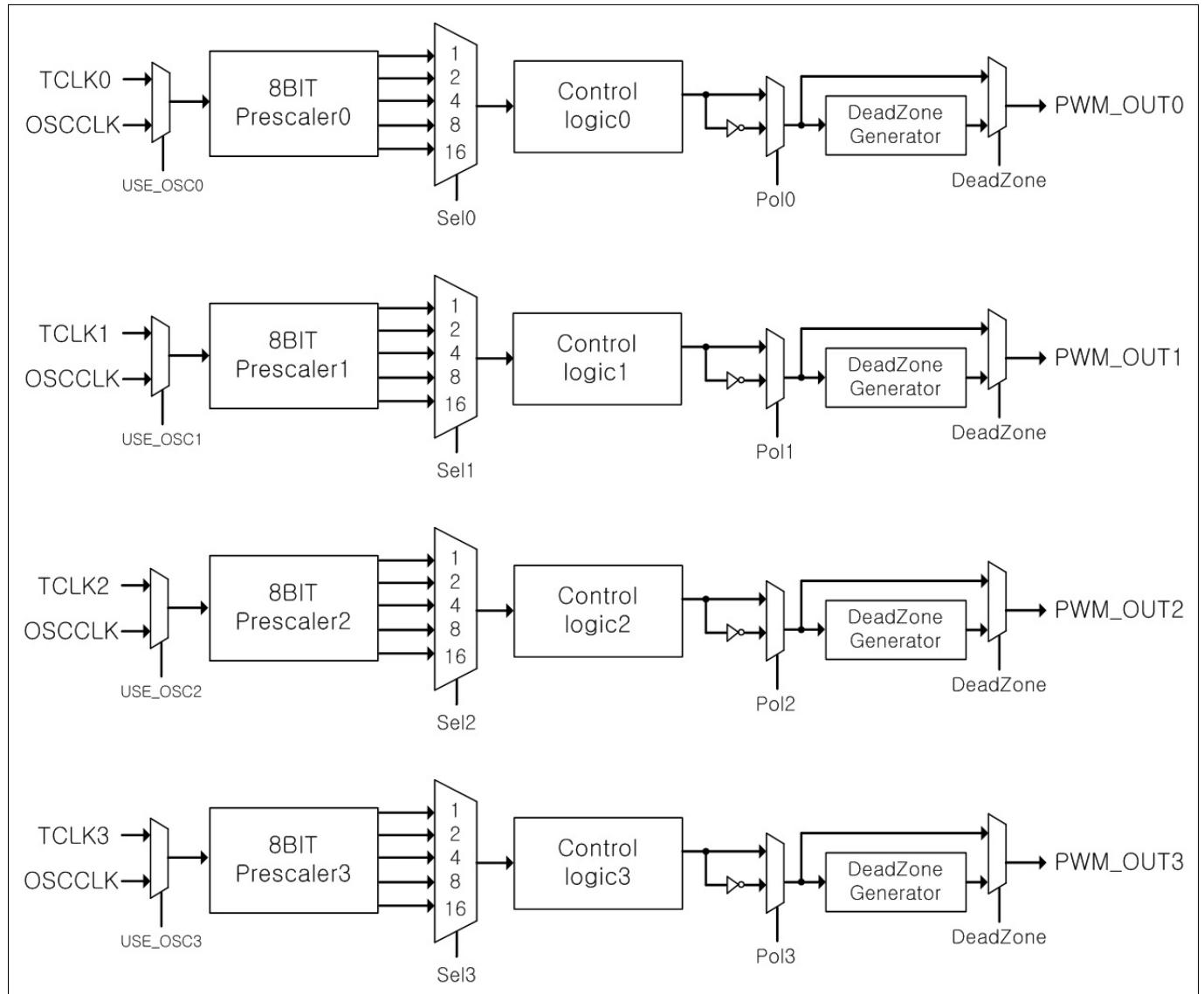


Figure 21.1: PWM Cycle

Steps in PWM Cycle:

- Initialize the TCNTBn register with 159 (50 + 109) and TCMPBn with 109.
- Start Timer: Sets the start bit and manually updates this bit to OFF.  
The TCNTBn value of 159 is loaded into the down-counter. Then, the output TOUTn is set to low.
- When down-counter counts down the value from TCNTBn to value in the TCMPBn register 109, the output changes from low to high.
- When the down-counter reaches 0, it generates an interrupt request.
- The down-counter automatically reloads TCNTBn. This restarts the cycle.



**Figure 21.2: PWMTIMER Clock Tree Diagram**

The Figure 21.2 shows the clock generation scheme for individual PWM Channels.

Each Timer can generate level interrupts.

## 21.2 Features

PWM supports these features:

- Four 32-bit Timers
- Four 8-bit Clock Prescalers that provide first level of division for the TCLK or the Oscillator Clock, and four Clock Dividers and Multiplexers that provide second level of division for the Prescaler clock
- To use Oscillator Clock as source clock, User must receive USE\_OSC signal from SYSREG outside PWM IP
- Programmable Clock Select Logic for individual PWM Channels
- Four Independent PWM Channels with Programmable Duty Control and Polarity
- Static Configuration: PWM is stopped
- Dynamic Configuration: PWM is running
- Auto-Reload and One-Shot Pulse Mode
- Dead Zone Generator on all PWM Outputs
- Level Interrupt Generation

The PWM has two modes of operation:

- Auto-Reload Mode: In this mode, continuous PWM pulses are generated based on Programmed Duty Cycle and Polarity.
- One-Shot Pulse Mode: In this mode, only one PWM pulse is generated based on Programmed Duty Cycle and Polarity.

To control the functionality of PWM, special function registers are provided. The PWM is a programmable output and a clock input AMBA slave module. It connects to the Advanced Peripheral Bus (APB). These special function registers within PWM are accessed through APB transactions.

## 21.3 PWM Operation

### 21.3.1 Prescaler and Divider

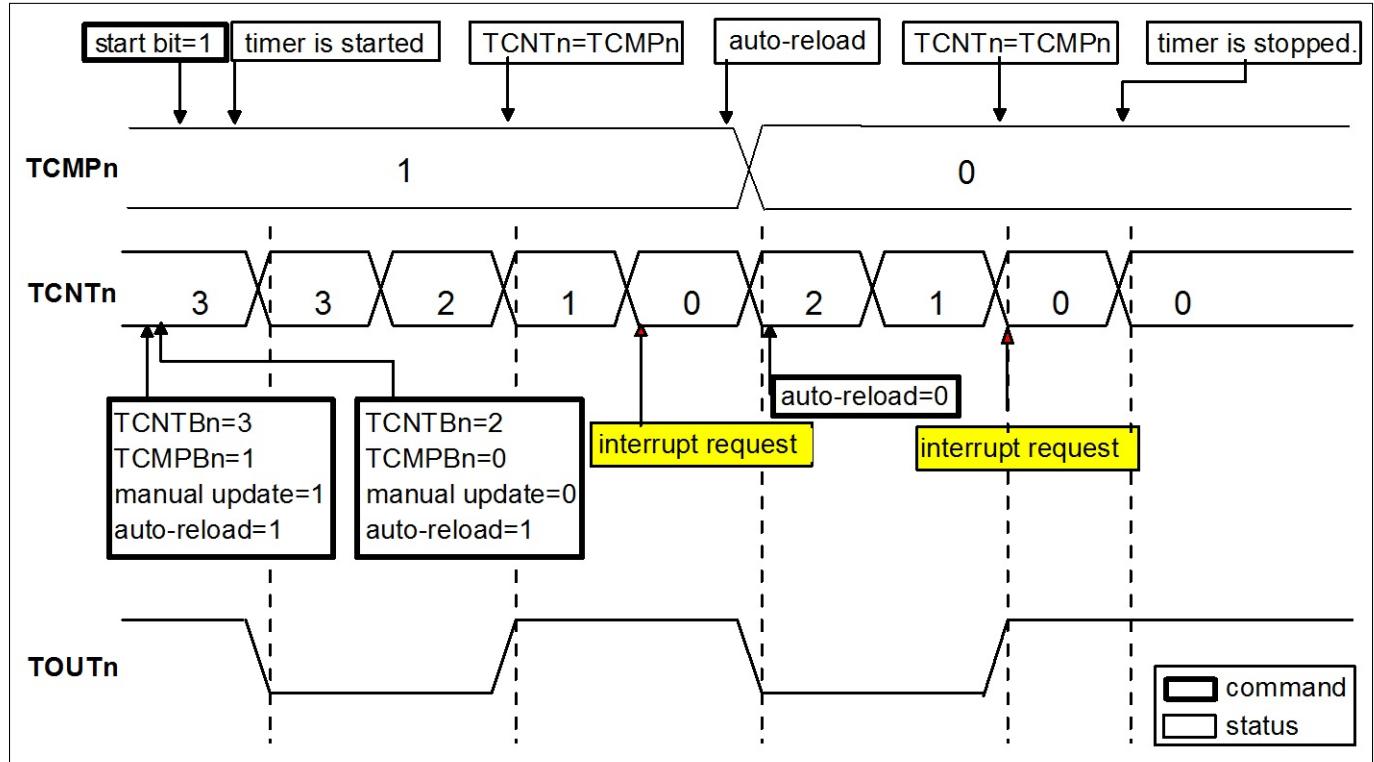
An 8-bit prescaler and 3-bit divider generates these output frequencies. Table 21.1 lists the minimum and maximum resolution based on Prescaler and Clock Divider values:

**Table 21.1:** Minimum and Maximum Resolution based on Prescaler and Clock Divider Values

4-bit Divider Settings	Minimum Resolution (Prescaler Value = 1)	Maximum Resolution (Prescaler Value = 255)	Maximum Interval (TCNTBn = 4294967295)
1/1(TCLK = 66 MHz)	0.030 us(33.0 MHz)	3.879 us(257.8 kHz)	16659.27 s
1/2(TCLK = 66 MHz)	0.061 us (16.5 MHz)	7.758 us (128.9 kHz)	33318.53 s
1/4(TCLK = 66 MHz)	0.121 us (8.25 MHz)	15.515 us (64.5 kHz)	66637.07 s
1/8(TCLK = 66 MHz)	0.242 us (4.13 MHz)	31.03 us (32.2 kHz)	133274.14 s
1/16(TCLK = 66 MHz)	0.485 us (2.06 MHz)	62.061 us (16.1 kHz)	266548.27 s

### 21.3.2 Basic Timer Operation

Figure 21.3 illustrates the Basic Timer Operation:



**Figure 21.3:** Timer Operations

The Timer comprises of four registers: TCNTBn, TCNTn, TCMPBn and TCMPn. When the Timer reaches 0, the TCNTBn and TCMPBn registers are loaded into TCNTn and TCMPn. When TCNTn reaches 0, the interrupt request occurs if the interrupt is enabled. TCNTn and TCMPn are the names of the internal registers. The TCNTn register is read from the TCNTOn register.

To generate interrupt at intervals three-cycle of XpwmTOUTn, set TCNTBn, TCMPBn and TCON.

Steps to generate interrupt:

1. Set TCNTBn = 3 and TCMPBn = 1.
2. Set auto-reload = 1 and manual update = 1.

When manual update bit is 1, the TCNTBn and TCMPBn values are loaded to TCNTn and TCMPn.

3. Set TCNTBn = 2 and TCMPBn = 0 for the next operation.

4. Set auto-reload = 1 and manual update = 0.

If you set manual update = 1, TCNTn is changed to 2 and TCMP is changed to 0.

Therefore, interrupt is generated at interval two-cycle instead of three-cycle.

Set auto-reload = 1 automatically, for the next operation.

5. Set start = 1 for starting the operation. Then, TCNTn is down counting.

When TCNTn is 0, interrupt is generated and if auto-reload is enable, TCNTn is loaded 2 (TCNTBn value) and TCMPn is loaded 0 (TCMPn value).

6. TCNTn is down counting before it stops.

### 21.3.3 Auto-reload and Double Buffering

The PWM Timers includes a double buffering feature, which changes the reload value for the next Timer operation without stopping the current Timer operation.

The Timer value is written into TCNTBn (Timer Count Buffer register) and the current counter value of the Timer is read from TCNTOn (Timer Count Observation register). If TCNTBn is Read, the Read value does not reflect the current state of the counter. It reflects the reload value for the next Timer duration.

Auto-reload copies the TCNTBn into TCNTn, when TCNTn reaches 0. The value written to TCNTBn, is loaded to TCNTn when the TCNTn reaches 0 and auto-reload is enabled. When the TCNTn is 0 and the auto-reload bit is 0, the TCNTn does not operate further.

Figure 21.4 illustrates an example of Double-Buffering:

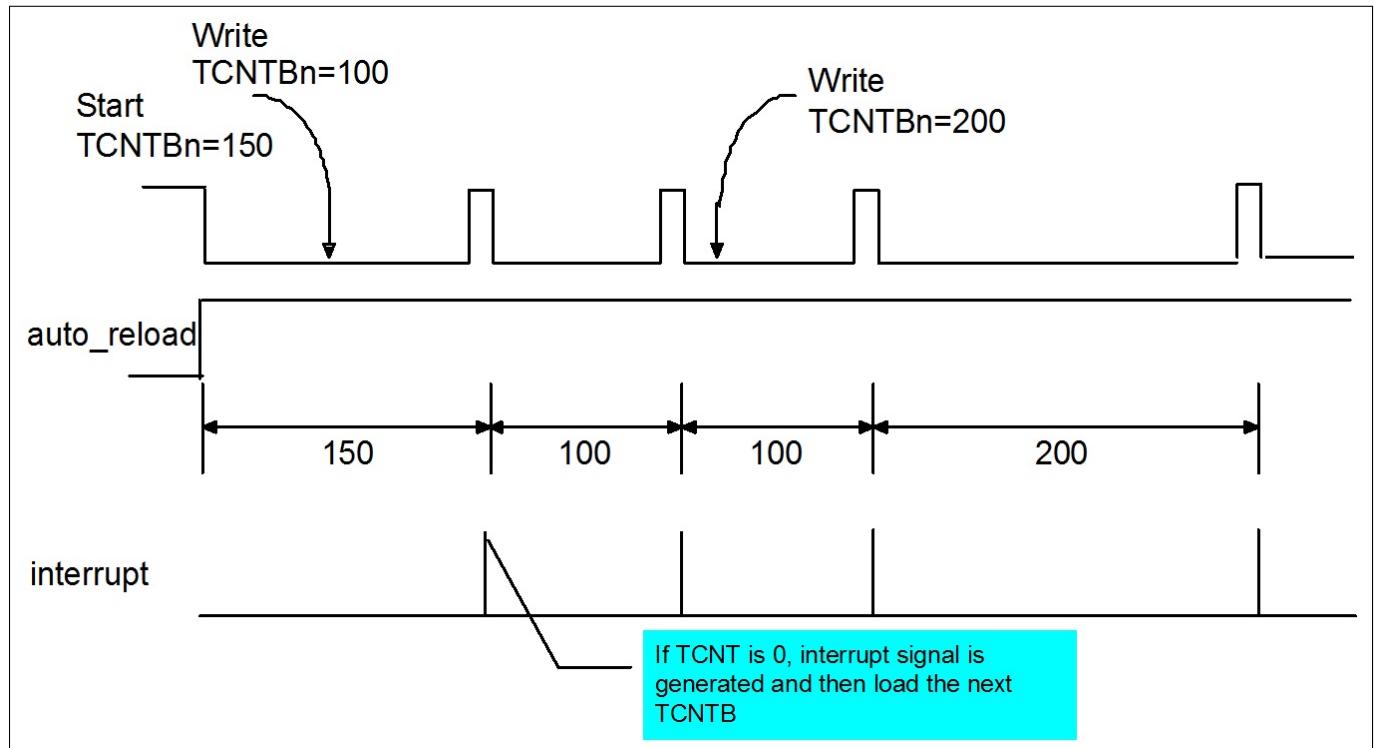


Figure 21.4: Double Buffering

### 21.3.4 Timer Operation

Figure 21.5 illustrates an example of Timer Operation:

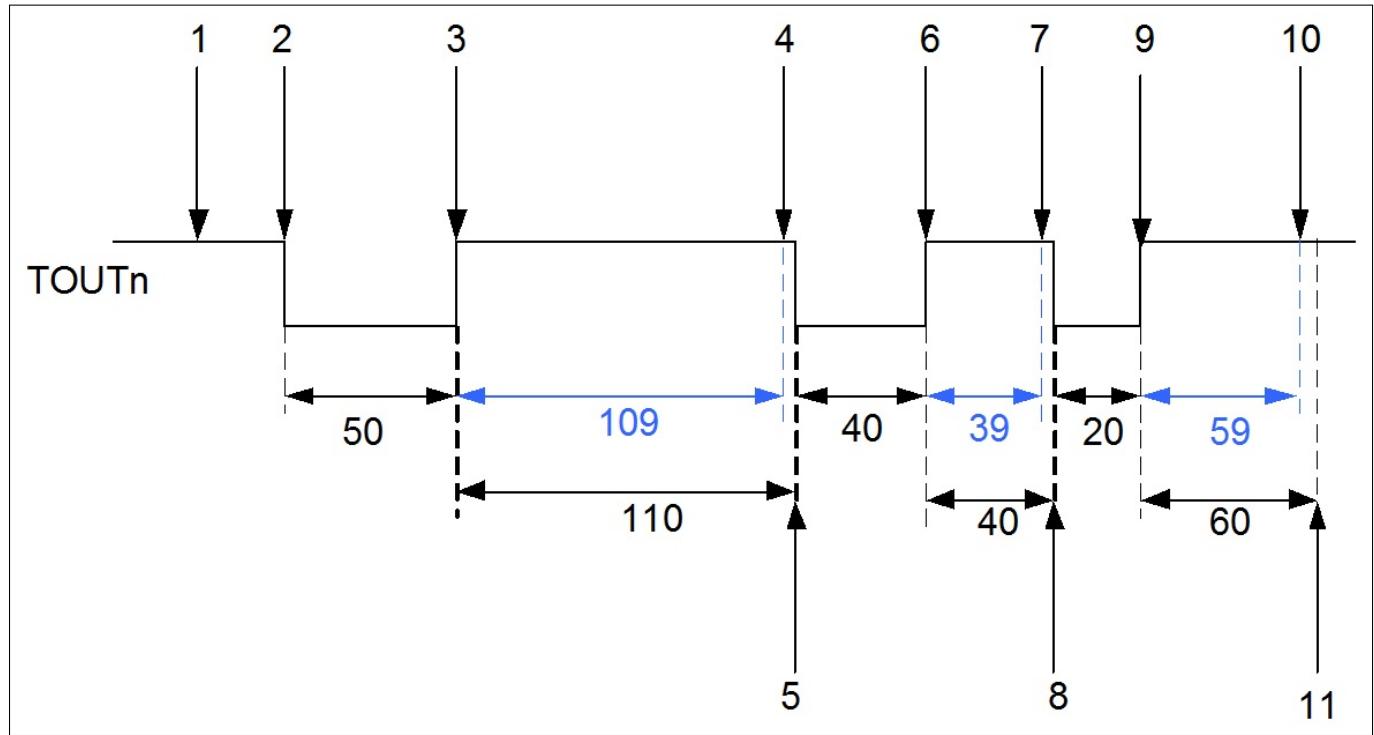


Figure 21.5: Timer Operation

Steps in Timer Operation:

1. Enable the Auto-reload feature. Set the TCNTBn as 159 (50 + 109) and TCMPBn as 109. Set the manual update bit On and set the manual update bit Off. Set the inverter On/ Off bit. The manual update bit sets TCNTn and TCMPn to the value of TCNTBn and TCMPBn.
2. Set TCNTBn and TCMPBn as 79 (40 + 39) and 39, respectively.
3. Start Timer: Set the start bit in TCON.
4. When TCNTn and TCMPn have the same value, the logic level of TOUTn is changed from low to high.
5. When TCNTn reaches 0, it generates interrupt request.
6. TCNTn and TCMPn are automatically reloaded with TCNTBn and TCMPBn as (79(40 + 39)) and 39, respectively.

In the Interrupt Service Routine (ISR), the TCNTBn and TCMPBn are set as 79 (20 + 59) and 59, respectively.

7. When TCNTn and TCMPn have the same value, the logic level of TOUTn is changed from low to high
8. When TCNTn reaches 0, it generates interrupt request.
9. TCNTn and TCMPn are automatically reloaded with TCNTBn and TCMPBn as (79(20 + 59)) and 59, respectively.

The Auto-reload and interrupt request are disabled to stop the Timer in the ISR.

10. When TCNTn and TCMPn have the same value, the logic level of TOUTn is changed from low to high
11. Even if TCNTn reaches 0, no interrupt request is generated.

12. TCNTn is not reloaded and the Timer is stopped because Auto-reload is disabled.

### 21.3.5 Initialize Timer (Setting Manual-Up Data and Inverter)

User must define the starting value of the TCNTn, because an Auto-reload operation of the Timer occurs when the down counter reaches 0. In this case, the starting value must be loaded by manual update bit.

The sequence to start a Timer is:

1. Write the initial value into TCNTBn and TCMPBn.
2. Set the manual update bit and clear only manual update bit of the corresponding Timer.  
NOTE: It is recommended to set the inverter On/Off bit (whether inverter is used or not).
3. Set the start bit of the corresponding Timer to start the Timer.

### 21.3.6 PWM (Pulse Width Modulation)

Figure 21.6 illustrates an example of PWM:

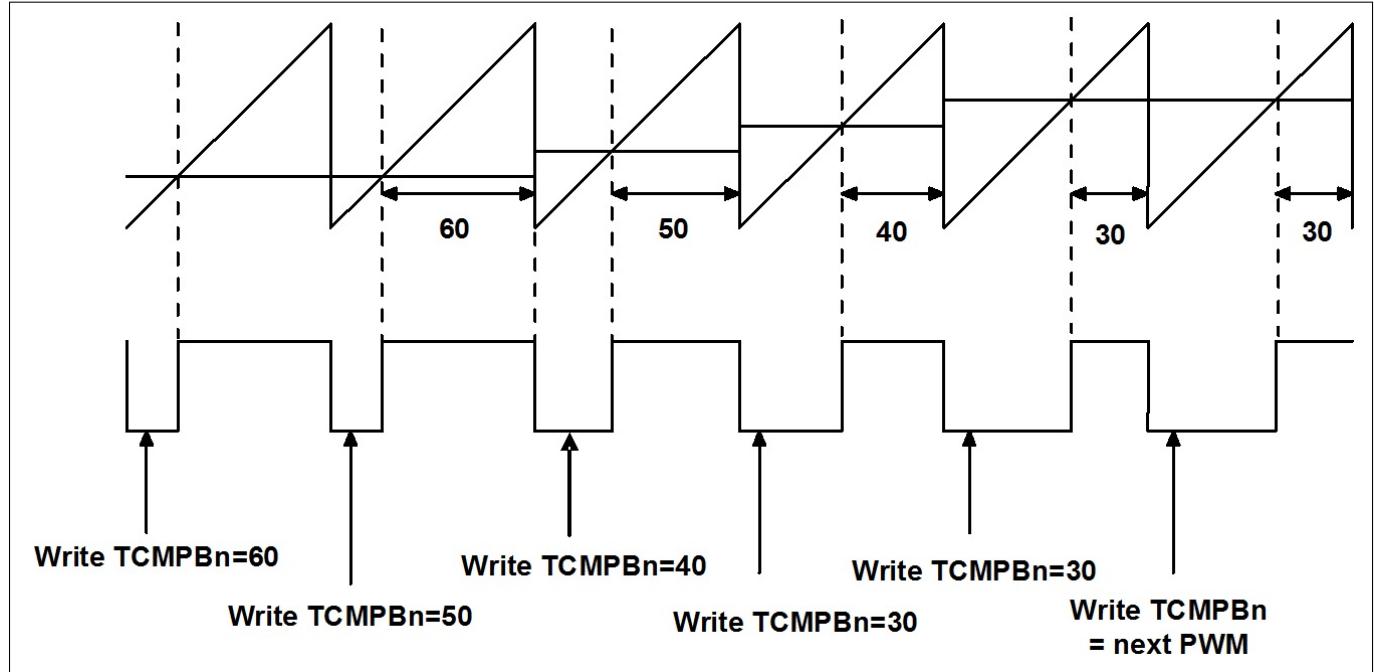


Figure 21.6: PWM

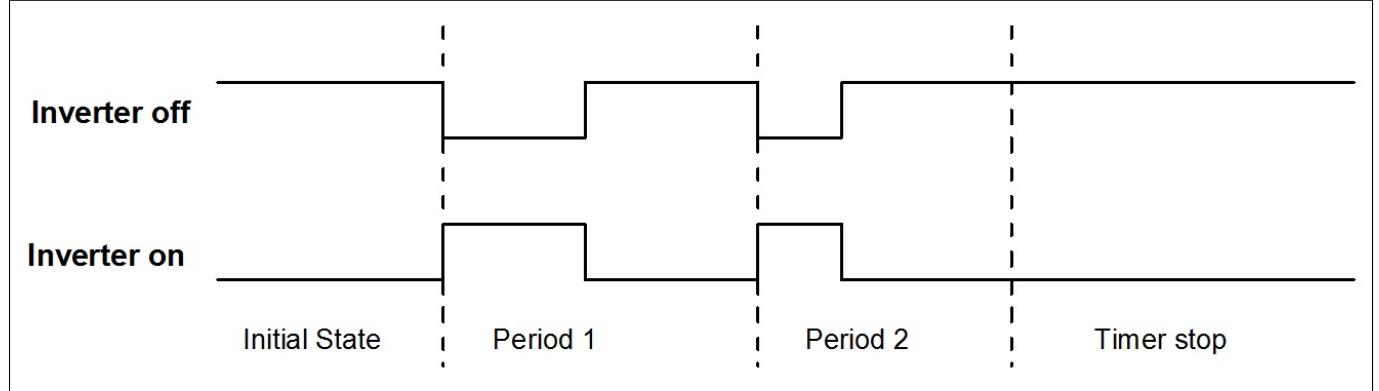
Use TCMPBn to implement the Pulse Width Modulation (PWM) feature. PWM frequency is determined by TCNTBn. A PWM value is determined by TCMPBn as shown in the Figure 21.6.

For a higher PWM value, decrease the TCMPBn value. For a lower PWM value, increase the TCMPBn value. When the output inverter is enabled, the increment/ decrement can be reverse.

Due to the Double Buffering feature, TCMPBn, for a next PWM cycle is written by ISR at any point of current PWM cycle.

### 21.3.7 Output Level Control

Figure 21.7 illustrates an example of inverter On/Off:



**Figure 21.7:** Inverter On/Off

Steps to maintain TOUT as high or low (assume that inverter is off):

1. Turn-off the Auto-reload bit. The TOUTn goes to high level and the Timer is stopped after TCNTn reaches 0. This method is recommended.
2. Stop the timer by clearing the timer start/stop bit to 0. When TCNTn <= TCMPn, the output level is high. When TCNTn > TCMPn, the output level is low.
3. TOUTn is inverted by the inverter on/off bit in TCON. The inverter removes the additional circuit to adjust the output level.

### 21.3.8 Dead Zone Generator

This feature inserts the time gap between a Turn-off and Turn-on of two different switching devices. This time gap prohibits the two switching device turning on simultaneously, even for a very short time.

TOUT\_0 specifies the PWM output. nTOUT\_0 specifies the inversion of the TOUT\_0. When the dead-zone is enabled, the output wave-form of TOUT\_0 and nTOUT\_0 is TOUT\_0\_DZ and nTOUT\_0\_DZ, respectively. TOUT0\_DZ and nTOUT\_0\_DZ cannot be turned on simultaneously by the dead zone interval. For functional correctness, the dead zone length must be set smaller than compare counter value.

Figure 21.8 illustrates the waveform when a dead zone feature is enabled:

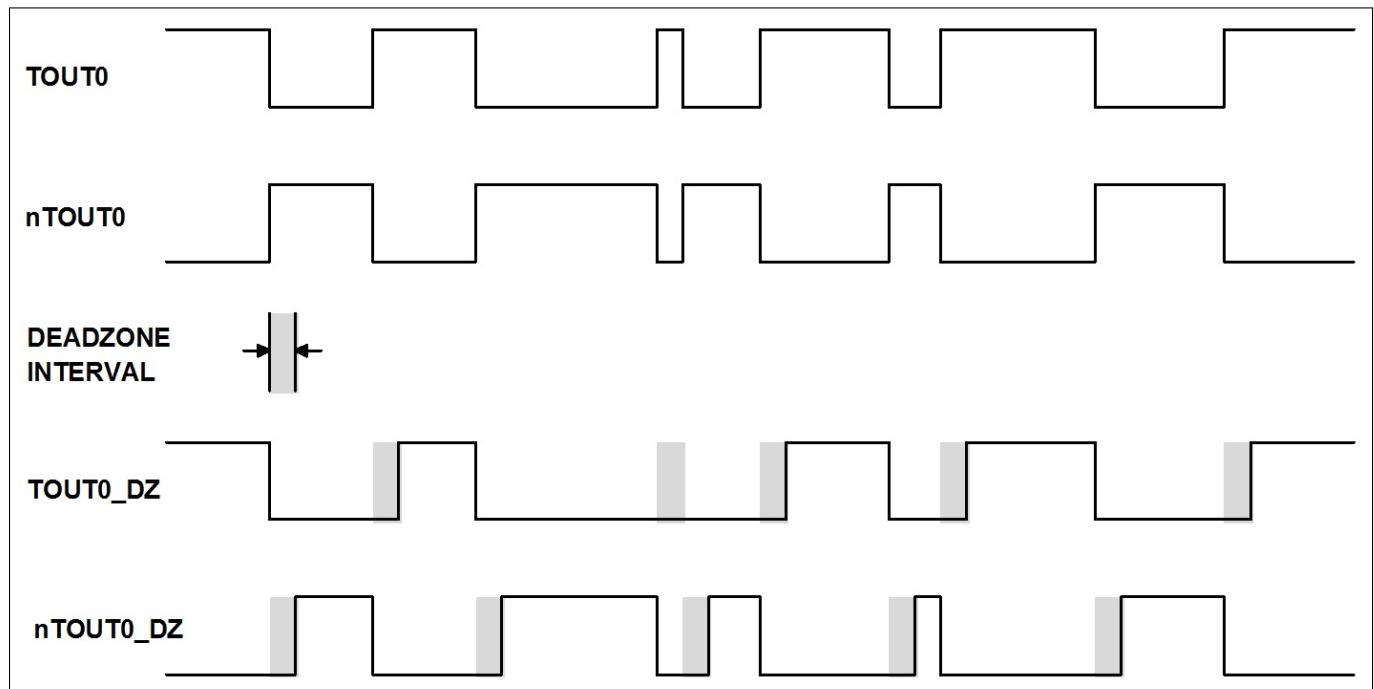


Figure 21.8: Waveform when a Dead Zone Feature is Enabled

## 21.4 I/O Description

Signal	I/O	Description	Pad	Type
TOUT_0	Output	PWMTIMER TOUT[0]	XpwmTOUT[0]	muxed
TOUT_1	Output	PWMTIMER TOUT[1]	XpwmTOUT[1]	muxed
TOUT_2	Output	PWMTIMER TOUT[2]	XpwmTOUT[2]	muxed
TOUT_3	Output	PWMTIMER TOUT[3]	XpwmTOUT[3]	muxed

NOTE: Type field indicates whether pads are dedicated to the signal, or pads are connected to the multiplexed signals.

## 21.5 PWM Register Description

### 21.5.1 Register Map Summary

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])

Register	Offset	Description	Reset Value
T0CFG0	0x00	PWMTIMER 0 CONFIGURATION REGISTER 0	0x0000_0000
T0CFG1	0x04	PWMTIMER 0 CONFIGURATION REGISTER 1	0x0000_0000
T0CON	0x08	PWMTIMER 0 CONTROL REGISTER	0x0000_0000
T0CMPB	0x10	PWMTIMER 0 COMPARE BUFFER REGISTER	0x0000_0000
T0CNTO	0x14	PWMTIMER 0 COUNT OBSERVATION REGISTER	0x0000_0000
T0INT_CSTAT	0x18	PWMTIMER 0 INTERRUPT CONTROL AND STATUS REGISTER	0x0000_0000
T0CFG0	0x100	PWMTIMER 1 CONFIGURATION REGISTER 0	0x0000_0000
T0CFG1	0x104	PWMTIMER 1 CONFIGURATION REGISTER 1	0x0000_0000
T0CON	0x108	PWMTIMER 1 CONTROL REGISTER	0x0000_0000
T0CMPB	0x110	PWMTIMER 1 COMPARE BUFFER REGISTER	0x0000_0000
T0CNTO	0x114	PWMTIMER 1 COUNT OBSERVATION REGISTER	0x0000_0000
T0INT_CSTAT	0x118	PWMTIMER 1 INTERRUPT CONTROL AND STATUS REGISTER	0x0000_0000
T0CFG0	0x200	PWMTIMER 2 CONFIGURATION REGISTER 0	0x0000_0000
T0CFG1	0x204	PWMTIMER 2 CONFIGURATION REGISTER 1	0x0000_0000
T0CON	0x208	PWMTIMER 2 CONTROL REGISTER	0x0000_0000
T0CMPB	0x210	PWMTIMER 2 COMPARE BUFFER REGISTER	0x0000_0000
T0CNTO	0x214	PWMTIMER 2 COUNT OBSERVATION REGISTER	0x0000_0000
T0INT_CSTAT	0x218	PWMTIMER 2 INTERRUPT CONTROL AND STATUS REGISTER	0x0000_0000
T0CFG0	0x300	PWMTIMER 3 CONFIGURATION REGISTER 0	0x0000_0000
T0CFG1	0x304	PWMTIMER 3 CONFIGURATION REGISTER 1	0x0000_0000
T0CON	0x308	PWMTIMER 3 CONTROL REGISTER	0x0000_0000
T0CMPB	0x310	PWMTIMER 3 COMPARE BUFFER REGISTER	0x0000_0000
T0CNTO	0x314	PWMTIMER 3 COUNT OBSERVATION REGISTER	0x0000_0000
T0INT_CSTAT	0x318	PWMTIMER 3 INTERRUPT CONTROL AND STATUS REGISTER	0x0000_0000

### 21.5.2 T0CFG0

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x00 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
PWMTIMER_0_Dead_zone_length	[23:16]	R/W	Dead Zone Length	0
RSVD	[15:8]	-	Reserved	-
PWMTIMER_0_Prescaler	[7:0]	R/W	Prescaler value	1

### 21.5.3 T0CFG1

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
PWMTIMER_0_Divider_MUX	[2:0]	R/W	Selects MUX input for PWM Timer 000 = 1/1 001 = 1/2 010 = 1/4 011 = 1/8 100 = 1/16	0

### 21.5.4 T0CON

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_0_By-pass_TCNT_TCOM	[5]	R/W	0 = normal PWM function 1 = Independently of the TCNTB / TCOMB value, the mux output from the Devider Mux is directly exported to TOUT.	0
PWMTIMER_0_Dead_zone_enable_disable	[4]	R/W	Dead Zone Generator Enable/Disable	0
PWMTIMER_0_auto_reload_on_off	[3]	R/W	0 = One-Shot 1 = Interval Mode (Auto-Reload)	0
PWMTIMER_0_output_inverter_on_off	[2]	R/W	0 = Inverter Off 1 = TOUT Inverter-On	0
PWMTIMER_0_manual_update	[1]	R/W	0 = No Operation 1 = Updates TCNTB0,TCMPB0	0
PWMTIMER_0_count_buffer	[31:0]	R/W	Count Buffer register	0
PWMTIMER_0_start_stop	[0]	R/W	0 = Stops 1 = Starts Timer 0	0

### 21.5.5 T0CMPB

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x10 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_0_compare_buffer	[31:0]	R/W	Compare Buffer register	0

### 21.5.6 T0CNOT

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x14 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_0_count_observation	[31:0]	R/W	Count Observation register	0

### 21.5.7 T0INT\_CSTAT

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x18 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_0_interrupt_status	[5]	R/W	Timer Interrupt Status bit Clears by writing 1 on this bit.	0
RSVD	[4:1]	-	Reserved	-
PWMTIMER_0_interrupt_enable	[0]	R/W	Enables Timer Interrupt 1 = Enables interrupt 0 = Disables interrupt	0

### 21.5.8 T0CFG0

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x100 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
PWMTIMER_1_Dead_zone_length	[23:16]	R/W	Dead Zone Length	0
RSVD	[15:8]	-	Reserved	-
PWMTIMER_1_Prescaler	[7:0]	R/W	Prescaler value	1

### 21.5.9 T0CFG1

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x104 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
PWMTIMER_1_Di- vider_MUX	[2:0]	R/W	Selects MUX input for PWM Timer 000 = 1/1 001 = 1/2 010 = 1/4 011 = 1/8 100 = 1/16	0

### 21.5.10 T0CON

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x108 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_1_By- pass_TCNT_TCOM	[5]	R/W	0 = normal PWM function 1 = Independently of the TCNTB / TCOMB value, the mux output from the Devider Mux is directly exported to TOUT.	0
PWMTIMER_1_Dead_ zone_enable_disable	[4]	R/W	Dead Zone Generator Enable/Disable	0
PWMTIMER_1_auto_ reload_on_off	[3]	R/W	0 = One-Shot 1 = Interval Mode (Auto-Reload)	0
PWMTIMER_1_output_ inverter_on_off	[2]	R/W	0 = Inverter Off 1 = TOUT Inverter-On	0
PWMTIMER_1_man- ual_update	[1]	R/W	0 = No Operation 1 = Updates TCNTB0,TCMPB0	0
PWMTIMER_1_start_ stop	[0]	R/W	0 = Stops 1 = Starts Timer 0	0

### 21.5.11 T0CMPB

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x110 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_1_com- pare_buffer	[31:0]	R/W	Compare Buffer register	0

### 21.5.12 T0CINTO

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x114 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_1_count_observation	[31:0]	R/W	Count Observation register	0

### 21.5.13 T0INT\_CSTAT

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x118 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_1_interrupt_status	[5]	R/W	Timer Interrupt Status bit Clears by writing 1 on this bit.	0
RSVD	[4:1]	-	Reserved	-
PWMTIMER_1_interrupt_enable	[0]	R/W	Enables Timer Interrupt 1 = Enables interrupt 0 = Disables interrupt	0

### 21.5.14 T0CFG0

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x200 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-

PWMTIMER_2_Dead_zone_length	[23:16]	R/W	Dead Zone Length	0
RSVD	[15:8]	-	Reserved	-
PWMTIMER_2_Prescaler	[7:0]	R/W	Prescaler value	1

### 21.5.15 T0CFG1

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x204 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
PWMTIMER_2_Divider_MUX	[2:0]	R/W	Selects MUX input for PWM Timer 000 = 1/1 001 = 1/2 010 = 1/4 011 = 1/8 100 = 1/16	0

### 21.5.16 T0CON

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x208 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_2_By-pass_TCNT_TCOM	[5]	R/W	0 = normal PWM function 1 = Independently of the TCNTB / TCOMB value, the mux output from the Devider Mux is directly exported to TOUT.	0
PWMTIMER_2_Dead_zone_enable_disable	[4]	R/W	Dead Zone Generator Enable/Disable	0
PWMTIMER_2_auto_reload_on_off	[3]	R/W	0 = One-Shot 1 = Interval Mode (Auto-Reload)	0
PWMTIMER_2_output_inverter_on_off	[2]	R/W	0 = Inverter Off 1 = TOUT Inverter-On	0
PWMTIMER_2_manual_update	[1]	R/W	0 = No Operation 1 = Updates TCNTB0,TCMPB0	0
PWMTIMER_2_start_stop	[0]	R/W	0 = Stops 1 = Starts Timer 0	0

### 21.5.17 T0CMPB

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x210 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_2_compare_buffer	[31:0]	R/W	Compare Buffer register	0

### 21.5.18 T0CNOTO

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x214 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_2_count_observation	[31:0]	R/W	Count Observation register	0

### 21.5.19 T0INT\_CSTAT

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x218 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_2_interrupt_status	[5]	R/W	Timer Interrupt Status bit Clears by writing 1 on this bit.	0
RSVD	[4:1]	-	Reserved	-
PWMTIMER_2_interrupt_enable	[0]	R/W	Enables Timer Interrupt 1 = Enables interrupt 0 = Disables interrupt	0

### 21.5.20 T0CFG0

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x300 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	-	Reserved	-
PWMTIMER_3_Dead_zone_length	[23:16]	R/W	Dead Zone Length	0
RSVD	[15:8]	-	Reserved	-
PWMTIMER_3_Prescaler	[7:0]	R/W	Prescaler value	1

### 21.5.21 T0CFG1

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x304 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	-	Reserved	-
PWMTIMER_3_Divider_MUX	[2:0]	R/W	Selects MUX input for PWM Timer 000 = 1/1 001 = 1/2 010 = 1/4 011 = 1/8 100 = 1/16	0

### 21.5.22 T0CON

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x308 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_3_By-pass_TCNT_TCOM	[5]	R/W	0 = normal PWM function 1 = Independently of the TCNTB / TCOMB value, the mux output from the Devider Mux is directly exported to TOUT.	0
PWMTIMER_3_Dead_zone_enable_disable	[4]	R/W	Dead Zone Generator Enable/Disable	0
PWMTIMER_3_auto_reload_on_off	[3]	R/W	0 = One-Shot 1 = Interval Mode (Auto-Reload)	0
PWMTIMER_3_output_inverter_on_off	[2]	R/W	0 = Inverter Off 1 = TOUT Inverter-On	0
PWMTIMER_3_manual_update	[1]	R/W	0 = No Operation 1 = Updates TCNTB0,TCMPB0	0
PWMTIMER_3_start_stop	[0]	R/W	0 = Stops 1 = Starts Timer 0	0

### 21.5.23 T0CMPB

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x310 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_3_compare_buffer	[31:0]	R/W	Compare Buffer register	0

### 21.5.24 T0CNOT

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x314 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
PWMTIMER_3_count_observation	[31:0]	R/W	Count Observation register	0

### 21.5.25 T0INT\_CSTAT

- Base Address: 208E0000(PWM[0])
- Base Address: 208F0000(PWM[1])
- Base Address: 20900000(PWM[2])
- Base Address: 20910000(TIMER[0])
- Base Address: 20920000(TIMER[1])
- Address = Base Address + 0x318 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:6]	-	Reserved	-
PWMTIMER_3_interrupt_status	[5]	R/W	Timer Interrupt Status bit Clears by writing 1 on this bit.	0
RSVD	[4:1]	-	Reserved	-
PWMTIMER_3_interrupt_enable	[0]	R/W	Enables Timer Interrupt 1 = Enables interrupt 0 = Disables interrupt	0

# 22 WDT

## 22.1 Overview

Watchdog timer is used to resume the controller operation whenever it is disturbed by malfunction such as noise and system error. It can be used as normal 16bit interval timer to request interrupt service. The watchdog timer generates the reset signal.

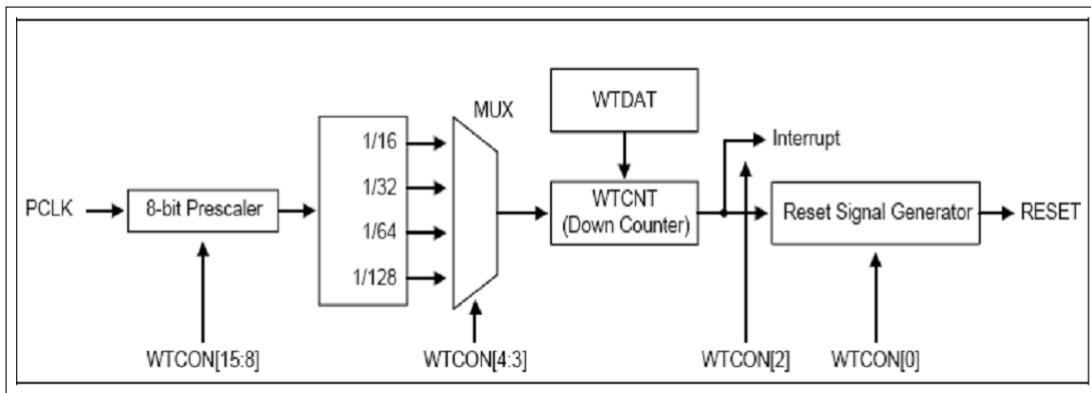
Difference in usage WDT compared with PWM timer is that WDT generates the reset signal.

DRONE\_SOC support 2 WDT, one is used to secure mode and the other is used to non-secure mode. Secure WDT generates the reset signal, non-secure WDT generates the interrupt request.

## 22.2 Functional Description

### 22.2.1 Watchdog Timer Operation

Feature 22.1 shows the functional block diagram of the watchdog timer. The watchdog timer uses only PCLK as its source clock. The PCLK frequency is prescaled to generate the corresponding watchdog timer clock, and the resulting frequency is divided again.



**Figure 22.1:** Watchdog Timer Block Diagram

The prescaler value and the frequency division factor are specified in the watchdog timer control (WDTCON) register. Valid prescaler values range from 0 to  $2^8$ -1. The frequency division factor can be selected as 16, 32, 64 or 128.

Use the following equation to calculate the watchdog timer clock frequency and the duration of each timer clock cycle :

$$t_{\text{watchdog}} = 1 / (\text{PCLK} / (\text{Prescaler value} + 1) / \text{Division\_factor})$$

### 22.2.2 WTDAT & WTCNT

Once the watchdog timer is enabled, the value of watchdog timer data (WTDAT) register cannot be automatically reloaded into the timer counter (WTCNT). In this reason, an initial value must be written to the watchdog timer count (WTCNT) register, before the watchdog timer starts.

Register	R/W	Description	Reset Value
WTCON	R/W	Watchdog timer control register	0x8021
WTDAT	R/W	Watchdog timer data register	0x8000
WTCNT	R/W	Watchdog timer count register	0x8000
WTCLRINT	W	Watchdog timer interrupt register	-

**Table 22.1:** WDT Mamory map

## 22.2.3 Special Function Register

### 22.2.3.1 Memory map

### 22.2.3.2 Watchdog timer control(WTCON) register

The WTCON register allows the user to enable/disable the watchdog timer, select the clock signal from 4 different sources, enable/disable interrupts, and enable/disable the watchdog timer output.

The Watchdog timer is used to resume restart in mal-function after its power on, if controller restart is not desired, the Watchdog timer should be disabled.

If the user wants to use the normal timer provided by the Watchdog timer, enable the interrupt and disable the Watchdog timer.

### 22.2.3.3 Watchdog timer data(WTDAT) register

The WTDAT register is used to specify the time-out duration. The content of WTDAT cannot be automatically loaded into the timer counter at initial watchdog timer operation. However, using 0x8000(initial value) will drive the first time\_out. In this case, the value of WTDAT will be automatically reloaded into WTCNT.

### 22.2.3.4 Watchdog timer count(WTCNT) register

The WTCNT register contains the current count values for the watchdog timer during normal operation. Note that the content of WTDAT register cannot be automatically loaded into the timer count register when the watchdog timer is enabled initially, so the WTCNT register must be set to initial value before enabling it.

### 22.2.3.5 Watchdog timer interrupt(WTCLRINT) register

The WTCLRINT register is used to clear the interrupt. Interrupt service routine is responsible for clearing the relevant interrupt after the interrupt service is completed. Writing any values on this register clears the interrupt. Reading this register is not allowed.

## 22.3 WDT Register Description

### 22.3.1 Register Map Summary

- Base Address: 206D0000 (WDT)

Register	Offset	Description	Reset Value
WTCON	0x00	Control register	0x0000_0021
WTDAT	0x04	Data register	0x0000_0000
WTCNT	0x08	Count register	0x0000_0000
WTCLRINT	0x0C	Interrupt register	0x0000_0000

### 22.3.2 WTCON

- Base Address: 206D0000 (WDT)
- Address = Base Address + 0x00 Reset Value = 0x0000\_0021

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
prescaler_value	[15:8]	R/W	Prescaler value. The valid range is from 0 to 0xFFFFFFFF.	0x80
RSVD	[7:6]	-	Reserved	-
watchdog_timer	[5]	R/W	Enable or disable bit of Watchdog timer. 0 = Disable 1 = Enable	1
clock_select	[4:3]	R/W	Determine the clock division factor. 00 = 16 01 = 32 10 = 64 11 = 128	0
interrupt_generation	[2]	R/W	Enable or disable bit of the interrupt. 0 = Disable 1 = Enable	0
RSVD	[1]	-	Reserved	-
Reset_en	[0]	R/W	Enable or disable bit of Watchdog timer output for reset signal. 1 = Assert reset signal of the artik310 at watchdog time-out. 0 = Disable the reset function of the watchdog timer.	1

### 22.3.3 WTDAT

- Base Address: 206D0000 (WDT)
- Address = Base Address + 0x04 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
wtdat	[15:0]	R/W	Watchdog timer count value for reload.	0

### 22.3.4 WTCNT

- Base Address: 206D0000 (WDT)
- Address = Base Address + 0x08 Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:16]	-	Reserved	-
WTCNT	[15:0]	R/W	The current value of the watchdog timer	0

### 22.3.5 WTCLRINT

- Base Address: 206D0000 (WDT)
- Address = Base Address + 0x0C Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
WTCLRINT	[31:0]	W	Write any values clears the interrupt	0