



# BL808

## 参考手册

*Version: 1.0*

*Copyright @ 2022*

[www.bouffalolab.com](http://www.bouffalolab.com)

# 目录

1 系统和存储器概述 . . . . .	32
1.1 系统架构 . . . . .	32
1.2 处理器 . . . . .	33
1.2.1 M0 处理器介绍 . . . . .	34
1.2.1.1 主要功能特点 . . . . .	34
1.2.1.2 扩展功能 . . . . .	35
1.2.1.3 实现标准 . . . . .	35
1.2.2 D0 处理器介绍 . . . . .	35
1.2.2.1 主要功能特点 . . . . .	35
1.2.2.2 扩展功能 . . . . .	36
1.2.2.3 实现标准 . . . . .	36
1.2.3 LP 处理器介绍 . . . . .	36
1.2.3.1 主要功能特点 . . . . .	36
1.2.3.2 实现标准 . . . . .	37
1.3 启动 . . . . .	37
1.4 内存 . . . . .	37
1.5 地址映射 . . . . .	37
1.6 中断源 . . . . .	39
2 复位和时钟 . . . . .	41
2.1 简介 . . . . .	41
2.2 复位管理 . . . . .	41
2.3 时钟源 . . . . .	43
3 GLB . . . . .	47
3.1 简介 . . . . .	47
3.2 功能描述 . . . . .	47
3.2.1 时钟管理 . . . . .	47

3.2.2	复位管理 . . . . .	47
3.2.3	总线管理 . . . . .	47
3.2.4	内存管理 . . . . .	48
4	GPIO . . . . .	49
4.1	简介 . . . . .	49
4.2	主要特征 . . . . .	49
4.3	GPIO 输入设置 . . . . .	49
4.4	GPIO 输出设置 . . . . .	50
4.4.1	普通输出模式 . . . . .	50
4.4.2	Set/Clear 输出模式 . . . . .	50
4.4.3	可编程输出模式 . . . . .	50
4.4.4	可编程 Set/Clear 输出模式 . . . . .	53
4.5	I/O FIFO . . . . .	54
4.6	I/O 中断 . . . . .	54
5	ADC . . . . .	56
5.1	简介 . . . . .	56
5.2	主要特点 . . . . .	56
5.3	功能描述 . . . . .	57
5.3.1	ADC 引脚和内部信号 . . . . .	58
5.3.2	ADC 通道 . . . . .	58
5.3.3	ADC 时钟 . . . . .	59
5.3.4	ADC 转换模式 . . . . .	60
5.3.5	ADC 结果 . . . . .	60
5.3.6	ADC 中断 . . . . .	61
5.3.7	ADC FIFO . . . . .	61
5.3.8	ADC 设置流程 . . . . .	62
5.3.9	VBAT 测量 . . . . .	62
5.3.10	TSEN 测量 . . . . .	63
5.4	寄存器描述 . . . . .	63
5.4.1	gpadc_config . . . . .	63
5.4.2	gpadc_dma_rdata . . . . .	65
5.4.3	gpadc_pir_train . . . . .	65
6	DAC . . . . .	67
6.1	简介 . . . . .	67
6.2	主要特点 . . . . .	67
6.3	功能描述 . . . . .	67
6.4	寄存器描述 . . . . .	68
6.4.1	gpdac_config . . . . .	69
6.4.2	gpdac_dma_config . . . . .	70

6.4.3	gpdac_dma_wdata . . . . .	71
6.4.4	gpdac_tx_fifo_status . . . . .	71
7	DMA . . . . .	72
7.1	简介 . . . . .	72
7.2	主要特征 . . . . .	72
7.3	功能描述 . . . . .	73
7.3.1	工作原理 . . . . .	73
7.3.2	DMA 通道配置 . . . . .	74
7.3.3	外设支持 . . . . .	74
7.3.4	链表模式 . . . . .	76
7.3.5	DMA 中断 . . . . .	77
7.4	传输模式 . . . . .	77
7.4.1	内存到内存 . . . . .	77
7.4.2	内存到外设 . . . . .	78
7.4.3	外设到内存 . . . . .	78
7.5	寄存器描述 . . . . .	78
7.5.1	DMA_IntStatus . . . . .	81
7.5.2	DMA_IntTCStatus . . . . .	81
7.5.3	DMA_IntTCClear . . . . .	81
7.5.4	DMA_IntErrorStatus . . . . .	82
7.5.5	DMA_IntErrClr . . . . .	82
7.5.6	DMA_RawIntTCStatus . . . . .	83
7.5.7	DMA_RawIntErrorStatus . . . . .	83
7.5.8	DMA_EnbldChns . . . . .	83
7.5.9	DMA_SoftBReq . . . . .	84
7.5.10	DMA_SoftSReq . . . . .	84
7.5.11	DMA_SoftLBReq . . . . .	84
7.5.12	DMA_SoftLSReq . . . . .	85
7.5.13	DMA_Config . . . . .	85
7.5.14	DMA_Sync . . . . .	85
7.5.15	DMA_C0SrcAddr . . . . .	86
7.5.16	DMA_C0DstAddr . . . . .	86
7.5.17	DMA_C0LLI . . . . .	86
7.5.18	DMA_C0Control . . . . .	87
7.5.19	DMA_C0Config . . . . .	88
7.5.20	DMA_C0RSVD . . . . .	89
7.5.21	DMA_C1SrcAddr . . . . .	90
7.5.22	DMA_C1DstAddr . . . . .	90
7.5.23	DMA_C1LLI . . . . .	90

7.5.24	DMA_C1Control . . . . .	91
7.5.25	DMA_C1Config . . . . .	92
7.5.26	DMA_C1RSVD . . . . .	93
7.5.27	DMA_C2SrcAddr . . . . .	93
7.5.28	DMA_C2DstAddr . . . . .	94
7.5.29	DMA_C2LLI . . . . .	94
7.5.30	DMA_C2Control . . . . .	94
7.5.31	DMA_C2Config . . . . .	96
7.5.32	DMA_C2RSVD . . . . .	97
7.5.33	DMA_C3SrcAddr . . . . .	97
7.5.34	DMA_C3DstAddr . . . . .	97
7.5.35	DMA_C3LLI . . . . .	98
7.5.36	DMA_C3Control . . . . .	98
7.5.37	DMA_C3Config . . . . .	100
7.5.38	DMA_C3RSVD . . . . .	100
7.5.39	DMA_C4SrcAddr . . . . .	101
7.5.40	DMA_C4DstAddr . . . . .	101
7.5.41	DMA_C4LLI . . . . .	101
7.5.42	DMA_C4Control . . . . .	102
7.5.43	DMA_C4Config . . . . .	103
7.5.44	DMA_C4RSVD . . . . .	104
7.5.45	DMA_C5SrcAddr . . . . .	104
7.5.46	DMA_C5DstAddr . . . . .	105
7.5.47	DMA_C5LLI . . . . .	105
7.5.48	DMA_C5Control . . . . .	105
7.5.49	DMA_C5Config . . . . .	107
7.5.50	DMA_C5RSVD . . . . .	108
7.5.51	DMA_C6SrcAddr . . . . .	108
7.5.52	DMA_C6DstAddr . . . . .	108
7.5.53	DMA_C6LLI . . . . .	109
7.5.54	DMA_C6Control . . . . .	109
7.5.55	DMA_C6Config . . . . .	111
7.5.56	DMA_C6RSVD . . . . .	111
7.5.57	DMA_C7SrcAddr . . . . .	112
7.5.58	DMA_C7DstAddr . . . . .	112
7.5.59	DMA_C7LLI . . . . .	112
7.5.60	DMA_C7Control . . . . .	113
7.5.61	DMA_C7Config . . . . .	114
7.5.62	DMA_C7RSVD . . . . .	115

8 DMA2D . . . . .	116
8.1 简介 . . . . .	116
8.2 主要特征 . . . . .	116
8.3 功能描述 . . . . .	117
8.3.1 工作原理 . . . . .	117
8.3.2 图像平移 . . . . .	118
8.3.3 图像旋转 . . . . .	118
8.3.4 图像翻折 . . . . .	120
8.3.5 图像填充 . . . . .	121
8.3.6 Color Key . . . . .	122
8.4 寄存器描述 . . . . .	122
8.4.1 DMA2D_IntStatus . . . . .	123
8.4.2 DMA2D_IntTCStatus . . . . .	124
8.4.3 DMA2D_IntTCClear . . . . .	124
8.4.4 DMA2D_EnbldChns . . . . .	125
8.4.5 DMA2D_Config . . . . .	125
8.4.6 DMA2D_Sync . . . . .	125
8.4.7 DMA2D_SoftBReq . . . . .	126
8.4.8 DMA2D_SoftLBReq . . . . .	126
8.4.9 DMA2D_SoftSReq . . . . .	126
8.4.10 DMA2D_SoftLSReq . . . . .	127
8.4.11 DMA2D_C0SrcAddr . . . . .	127
8.4.12 DMA2D_C0DstAddr . . . . .	127
8.4.13 DMA2D_C0LLI . . . . .	128
8.4.14 DMA2D_C0_BUS . . . . .	128
8.4.15 DMA2D_C0_SRC_CNT . . . . .	129
8.4.16 DMA2D_C0_SRC_XIC . . . . .	130
8.4.17 DMA2D_C0_SRC_YIC . . . . .	130
8.4.18 DMA2D_C0_DST_CNT . . . . .	130
8.4.19 DMA2D_C0_DST_XIC . . . . .	131
8.4.20 DMA2D_C0_DST_YIC . . . . .	131
8.4.21 DMA2D_C0_KEY . . . . .	132
8.4.22 DMA2D_C0_KEY_EN . . . . .	132
8.4.23 DMA2D_C0_CFG . . . . .	133
9 LZ4D . . . . .	135
9.1 简介 . . . . .	135
9.2 主要特征 . . . . .	135
9.3 功能描述 . . . . .	135
9.4 时钟 . . . . .	136

9.5 编程流程 . . . . .	136
9.5.1 解压缩数据 . . . . .	136
9.6 寄存器描述 . . . . .	136
9.6.1 lz4_config . . . . .	137
9.6.2 lz4_src_fix . . . . .	137
9.6.3 lz4_dst_fix . . . . .	138
9.6.4 lz4_src_start . . . . .	138
9.6.5 lz4_src_end . . . . .	139
9.6.6 lz4_dst_start . . . . .	139
9.6.7 lz4_dst_end . . . . .	139
9.6.8 lz4_int_en . . . . .	140
9.6.9 lz4_int_sta . . . . .	140
9.6.10 lz4_monitor . . . . .	141
10 DBI . . . . .	142
10.1 简介 . . . . .	142
10.2 主要特点 . . . . .	142
10.3 功能描述 . . . . .	143
10.3.1 DBI Type B . . . . .	143
10.3.1.1 写时序 . . . . .	143
10.3.1.2 读时序 . . . . .	145
10.3.1.3 输出 RGB565 . . . . .	145
10.3.1.4 输出 RGB666 . . . . .	147
10.3.1.5 输出 RGB888 . . . . .	148
10.3.2 DBI Type C 3-Line . . . . .	149
10.3.2.1 写时序 . . . . .	149
10.3.2.2 读时序 . . . . .	149
10.3.2.3 输出 RGB565 . . . . .	150
10.3.2.4 输出 RGB666 . . . . .	151
10.3.2.5 输出 RGB888 . . . . .	151
10.3.3 DBI Type C 4-Line . . . . .	151
10.3.3.1 写时序 . . . . .	151
10.3.3.2 读时序 . . . . .	152
10.3.3.3 输出 RGB565 . . . . .	153
10.3.3.4 输出 RGB666 . . . . .	154
10.3.3.5 输出 RGB888 . . . . .	154
10.3.4 输入像素格式 . . . . .	155
10.3.5 中断 . . . . .	156
10.3.6 DMA . . . . .	156

10.4 寄存器描述 . . . . .	156
10.4.1 dbi_config . . . . .	157
10.4.2 dbi_int_sts . . . . .	158
10.4.3 dbi_bus_busy . . . . .	160
10.4.4 dbi_pix_cnt . . . . .	160
10.4.5 dbi_prd . . . . .	161
10.4.6 dbi_wdata . . . . .	161
10.4.7 dbi_rdata . . . . .	161
10.4.8 dbi_fifo_config_0 . . . . .	162
10.4.9 dbi_fifo_config_1 . . . . .	163
10.4.10 dbi_fifo_wdata . . . . .	163
11 DPI . . . . .	164
11.1 简介 . . . . .	164
11.2 主要特点 . . . . .	164
11.3 功能描述 . . . . .	165
11.3.1 DPI 时序 . . . . .	165
11.3.2 可编程定时参数 . . . . .	167
11.3.3 单缓冲区 . . . . .	168
11.3.4 乒乓缓冲区 . . . . .	168
11.3.5 测试模式 . . . . .	168
11.3.6 输入像素格式 . . . . .	169
11.3.6.1 YUV422(交织) . . . . .	170
11.3.6.2 RGB888 . . . . .	170
11.3.6.3 RGB565 . . . . .	171
11.3.6.4 RGBA8888 . . . . .	172
11.3.6.5 YUV420(平面) . . . . .	173
11.3.7 可编程中断 . . . . .	174
11.3.8 与 OSD 配合使用 . . . . .	175
12 DSI . . . . .	176
12.1 简介 . . . . .	176
12.2 主要特点 . . . . .	176
12.3 功能描述 . . . . .	177
12.3.1 DSI 协议分层 . . . . .	177
12.3.2 物理层 . . . . .	179
12.3.3 时钟通道 (Clock Lane) . . . . .	180
12.3.3.1 低功耗模式 (LPM, LP-11) . . . . .	181
12.3.3.2 超低功耗模式 (ULPM, LP-00) . . . . .	182
12.3.3.3 高速时钟模式 (HSCM, HS-0/1) . . . . .	183

12.3.4 数据通道 (Data Lanes) . . . . .	184
12.3.4.1 高速数据传输 (HSDT) . . . . .	184
12.3.4.2 Bus Turn-Around(BTA) . . . . .	186
12.3.4.3 Escape 模式 . . . . .	187
12.3.4.4 低功耗数据传输 (LPDT) . . . . .	188
12.3.4.5 超低功耗状态 (ULPS) . . . . .	190
12.3.4.6 远程复位 (RAR) . . . . .	190
12.3.4.7 应答 (ACK) . . . . .	191
12.3.5 通道管理层 . . . . .	192
12.3.6 协议层 . . . . .	195
12.3.6.1 短数据包 . . . . .	195
12.3.6.2 长数据包 . . . . .	196
12.3.6.3 每次传输多个数据包 . . . . .	197
12.3.6.4 数据包字节的位顺序 . . . . .	198
12.3.6.5 数据包的字节顺序 . . . . .	198
12.3.6.6 数据包头 (PH) . . . . .	199
12.3.6.7 数据标识 (DI) . . . . .	199
12.3.6.8 虚拟通道 (VC) . . . . .	199
12.3.6.9 数据类型 (DT) . . . . .	199
12.3.6.10 包数据 (PD) . . . . .	201
12.3.6.11 字计数 (WC) . . . . .	201
12.3.6.12 纠错码 (ECC) . . . . .	201
12.3.6.13 数据包尾 (PF) . . . . .	203
12.3.7 应用层 . . . . .	203
12.3.8 命令模式 . . . . .	204
12.3.9 视频模式 . . . . .	204
12.3.9.1 具有同步脉冲的非突发模式 (Non-Burst Mode with SYNC Pulses) . . . . .	204
12.3.9.2 具有同步事件的非突发模式 (Non-Burst Mode with SYNC Events) . . . . .	205
12.3.9.3 突发模式 (Burst Mode) . . . . .	205
12.3.10 Line Buffer . . . . .	206
12.3.11 显示数据格式 . . . . .	207
12.3.11.1 YUV422(8-bit) . . . . .	207
12.3.11.2 RGB565 . . . . .	208
12.3.11.3 RGB666(loosely packed) . . . . .	208
12.3.11.4 RGB888 . . . . .	209
12.3.12 中断 . . . . .	210
12.3.13 DMA . . . . .	211
12.3.14 与 OSD 配合使用 . . . . .	211

12.4 寄存器描述 . . . . .	211
12.4.1 dsi_config . . . . .	212
12.4.2 dsi_esc_config . . . . .	214
12.4.3 dsi_lpdt_tx_config . . . . .	214
12.4.4 dsi_hstx_config . . . . .	215
12.4.5 dsi_int_status . . . . .	216
12.4.6 dsi_int_mask . . . . .	216
12.4.7 dsi_int_clear . . . . .	217
12.4.8 dsi_int_enable . . . . .	218
12.4.9 dsi_fifo_config_0 . . . . .	218
12.4.10 dsi_fifo_config_1 . . . . .	219
12.4.11 dsi_fifo_wdata . . . . .	220
12.4.12 dsi_fifo_rdata . . . . .	220
12.4.13 dphy_config_0 . . . . .	221
12.4.14 dphy_config_1 . . . . .	222
12.4.15 dphy_config_2 . . . . .	223
12.4.16 dphy_config_3 . . . . .	223
12.4.17 dphy_config_4 . . . . .	224
12.4.18 dphy_config_5 . . . . .	224
12.4.19 dphy_config_6 . . . . .	225
12.4.20 dphy_config_7 . . . . .	225
12.4.21 dphy_config_8 . . . . .	227
12.4.22 dphy_config_9 . . . . .	228
12.4.23 dphy_config_10 . . . . .	228
12.4.24 dphy_config_11 . . . . .	230
12.4.25 dphy_config_12 . . . . .	230
12.4.26 dphy_config_13 . . . . .	230
12.4.27 dphy_config_14 . . . . .	231
12.4.28 dphy_config_15 . . . . .	231
12.4.29 dphy_config_16 . . . . .	232
12.4.30 dummy_reg . . . . .	233
13 CAM . . . . .	234
13.1 简介 . . . . .	234
13.2 主要特征 . . . . .	234
13.3 功能描述 . . . . .	235
13.3.1 DVP(Digital Video Port) 信号与配置 . . . . .	235
13.3.2 YCbCr 格式 . . . . .	235
13.3.3 输入源 . . . . .	236
13.3.4 影片模式/照片模式 . . . . .	237

13.3.5	图像矩形裁剪 . . . . .	237
13.3.6	行帧同步信号完整性检测 . . . . .	237
13.3.7	缓存图片信息 . . . . .	237
13.3.8	支持多种中断信息(可独立开关配置) . . . . .	238
13.4	寄存器描述 . . . . .	238
13.4.1	dvp2axi_configue . . . . .	239
13.4.2	dvp2axi_addr_start . . . . .	241
13.4.3	dvp2axi_mem_bcnt . . . . .	241
13.4.4	dvp_status_and_error . . . . .	241
13.4.5	dvp2axi_frame_bcnt . . . . .	243
13.4.6	dvp_frame_fifo_pop . . . . .	243
13.4.7	dvp2axi_frame_vld . . . . .	244
13.4.8	dvp2axi_frame_period . . . . .	244
13.4.9	dvp2axi_misc . . . . .	245
13.4.10	dvp2axi_hsync_crop . . . . .	245
13.4.11	dvp2axi_vsync_crop . . . . .	246
13.4.12	dvp2axi_fram_exm . . . . .	246
13.4.13	frame_start_addr0 . . . . .	246
13.4.14	frame_start_addr1 . . . . .	247
13.4.15	frame_start_addr2 . . . . .	247
13.4.16	frame_start_addr3 . . . . .	247
13.4.17	frame_id_sts01 . . . . .	248
13.4.18	frame_id_sts23 . . . . .	248
13.4.19	dvp_debug . . . . .	248
13.4.20	dvp_dummy_reg . . . . .	249
14	IR . . . . .	250
14.1	简介 . . . . .	250
14.2	主要特征 . . . . .	250
14.3	功能描述 . . . . .	250
14.3.1	固定协议接收 . . . . .	250
14.3.2	脉冲宽度接收 . . . . .	252
14.3.3	普通发送模式 . . . . .	252
14.3.4	脉冲宽度发送 . . . . .	252
14.3.5	载波调制 . . . . .	252
14.3.6	自由模式 . . . . .	252
14.3.7	DMA 模式 . . . . .	253
14.3.8	IR 中断 . . . . .	253
14.4	寄存器描述 . . . . .	253
14.4.1	irtx_config . . . . .	254

14.4.2	irtx_int_sts . . . . .	256
14.4.3	irtx_pulse_width . . . . .	257
14.4.4	irtx_pw_0 . . . . .	257
14.4.5	irtx_pw_1 . . . . .	258
14.4.6	irrx_config . . . . .	258
14.4.7	irrx_int_sts . . . . .	259
14.4.8	irrx_pw_config . . . . .	260
14.4.9	irrx_data_count . . . . .	260
14.4.10	irrx_data_word0 . . . . .	261
14.4.11	irrx_data_word1 . . . . .	261
14.4.12	irtx_fifo_config_0 . . . . .	261
14.4.13	irtx_fifo_config_1 . . . . .	262
14.4.14	ir_fifo_wdata . . . . .	263
14.4.15	ir_fifo_rdata . . . . .	263
15	SPI . . . . .	264
15.1	简介 . . . . .	264
15.2	主要特征 . . . . .	264
15.3	功能描述 . . . . .	265
15.3.1	时钟控制 . . . . .	265
15.3.2	主设备持续传输模式 . . . . .	266
15.3.3	主从设备传输接收数据 . . . . .	266
15.3.4	接收忽略功能 . . . . .	266
15.3.5	滤波功能 . . . . .	266
15.3.6	可配置 MSB/LSB 传输 . . . . .	267
15.3.7	可调整字节传输顺序 . . . . .	267
15.3.8	从模式超时机制 . . . . .	268
15.3.9	I/O 传输模式 . . . . .	268
15.3.10	DMA 传输模式 . . . . .	268
15.3.11	SPI 中断 . . . . .	268
15.4	寄存器描述 . . . . .	268
15.4.1	spi_config . . . . .	269
15.4.2	spi_int_sts . . . . .	271
15.4.3	spi_bus_busy . . . . .	272
15.4.4	spi_prd_0 . . . . .	273
15.4.5	spi_prd_1 . . . . .	273
15.4.6	spi_rxd_ignr . . . . .	274
15.4.7	spi_sto_value . . . . .	274
15.4.8	spi_fifo_config_0 . . . . .	274
15.4.9	spi_fifo_config_1 . . . . .	275

15.4.10 spi_fifo_wdata . . . . .	276
15.4.11 spi_fifo_rdata . . . . .	277
15.4.12 backup_io_en . . . . .	277
<b>16 UART . . . . .</b>	<b>278</b>
16.1 简介 . . . . .	278
16.2 主要特征 . . . . .	278
16.3 功能描述 . . . . .	279
16.3.1 数据格式描述 . . . . .	279
16.3.2 基本架构图 . . . . .	279
16.3.3 时钟源 . . . . .	280
16.3.4 波特率设定 . . . . .	280
16.3.5 滤波 . . . . .	281
16.3.6 发送器 . . . . .	282
16.3.7 接收器 . . . . .	282
16.3.8 自动波特率检测 . . . . .	282
16.3.9 硬件流控 . . . . .	283
16.3.10 DMA 传输模式 . . . . .	284
16.3.11 LIN 总线支持 . . . . .	284
16.3.12 RS485 模式 . . . . .	286
16.3.13 UART 中断 . . . . .	286
16.3.14 TX/RX 传输结束中断 . . . . .	286
16.3.15 TX/RX FIFO 请求中断 . . . . .	287
16.3.16 RX 超时中断 . . . . .	287
16.3.17 RX 奇偶校验错误中断 . . . . .	287
16.3.18 TX/RX FIFO 溢出中断 . . . . .	287
16.3.19 RX BCR 中断 . . . . .	287
16.3.20 LIN 同步错误中断 . . . . .	288
16.3.21 通用/固定字符模式自动波特率检测中断 . . . . .	288
16.4 寄存器描述 . . . . .	288
16.4.1 utx_config . . . . .	289
16.4.2 urx_config . . . . .	290
16.4.3 uart_bit_prd . . . . .	291
16.4.4 data_config . . . . .	291
16.4.5 utx_ir_position . . . . .	292
16.4.6 urx_ir_position . . . . .	292
16.4.7 urx_rto_timer . . . . .	293
16.4.8 uart_sw_mode . . . . .	293
16.4.9 uart_int_sts . . . . .	294
16.4.10 uart_int_mask . . . . .	295

16.4.11 uart_int_clear . . . . .	296
16.4.12 uart_int_en . . . . .	297
16.4.13 uart_status . . . . .	298
16.4.14 sts_urx_abr_prd . . . . .	298
16.4.15 urx_abr_prd_b01 . . . . .	299
16.4.16 urx_abr_prd_b23 . . . . .	299
16.4.17 urx_abr_prd_b45 . . . . .	299
16.4.18 urx_abr_prd_b67 . . . . .	300
16.4.19 urx_abr_pw_tol . . . . .	300
16.4.20 urx_bcr_int_cfg . . . . .	300
16.4.21 utx_rs485_cfg . . . . .	301
16.4.22 uart_fifo_config_0 . . . . .	301
16.4.23 uart_fifo_config_1 . . . . .	302
16.4.24 uart_fifo_wdata . . . . .	303
16.4.25 uart_fifo_rdata . . . . .	303
17 I2C . . . . .	304
17.1 简介 . . . . .	304
17.2 主要特征 . . . . .	304
17.3 功能描述 . . . . .	304
17.3.1 起始和停止条件 . . . . .	305
17.3.2 数据传输格式 . . . . .	305
17.3.3 仲裁 . . . . .	307
17.4 I2C 时钟设定 . . . . .	308
17.5 I2C 配置流程 . . . . .	308
17.5.1 配置项 . . . . .	308
17.5.2 读写标志位 . . . . .	309
17.5.3 从设备地址 . . . . .	309
17.5.4 从设备寄存器地址 . . . . .	309
17.5.5 从设备寄存器地址长度 . . . . .	309
17.5.6 数据 . . . . .	309
17.5.7 数据长度 . . . . .	309
17.5.8 使能信号 . . . . .	309
17.6 FIFO 管理 . . . . .	310
17.7 搭配使用 DMA . . . . .	310
17.7.1 DMA 发送流程 . . . . .	310
17.7.2 DMA 接收流程 . . . . .	311
17.8 中断 . . . . .	311
17.9 寄存器描述 . . . . .	311
17.9.1 i2c_config . . . . .	312

17.9.2	i2c_int_sts . . . . .	313
17.9.3	i2c_sub_addr . . . . .	314
17.9.4	i2c_bus_busy . . . . .	315
17.9.5	i2c_prd_start . . . . .	315
17.9.6	i2c_prd_stop . . . . .	316
17.9.7	i2c_prd_data . . . . .	316
17.9.8	i2c_fifo_config_0 . . . . .	317
17.9.9	i2c_fifo_config_1 . . . . .	318
17.9.10	i2c_fifo_wdata . . . . .	318
17.9.11	i2c_fifo_rdata . . . . .	319
18	PWM . . . . .	320
18.1	简介 . . . . .	320
18.2	主要特征 . . . . .	320
18.3	功能描述 . . . . .	320
18.3.1	时钟与分频器 . . . . .	320
18.3.2	有效电平 . . . . .	321
18.3.3	脉冲产生原理 . . . . .	321
18.3.4	刹车 . . . . .	322
18.3.5	死区 . . . . .	323
18.3.6	周期与占空比计算 . . . . .	323
18.3.7	PWM 中断 . . . . .	323
18.3.8	ADC 联动 . . . . .	323
18.4	寄存器描述 . . . . .	323
18.4.1	pwm_int_config . . . . .	324
18.4.2	pwm_mc0_config0 . . . . .	325
18.4.3	pwm_mc0_config1 . . . . .	327
18.4.4	pwm_mc0_period . . . . .	328
18.4.5	pwm_mc0_dead_time . . . . .	328
18.4.6	pwm_mc0_ch0_thre . . . . .	330
18.4.7	pwm_mc0_ch1_thre . . . . .	330
18.4.8	pwm_mc0_ch2_thre . . . . .	330
18.4.9	pwm_mc0_ch3_thre . . . . .	331
18.4.10	pwm_mc0_int_sts . . . . .	331
18.4.11	pwm_mc0_int_mask . . . . .	332
18.4.12	pwm_mc0_int_clear . . . . .	333
18.4.13	pwm_mc0_int_en . . . . .	334
18.4.14	pwm_mc1_config0 . . . . .	334
18.4.15	pwm_mc1_config1 . . . . .	336
18.4.16	pwm_mc1_period . . . . .	337

18.4.17 pwm_mc1_dead_time . . . . .	338
18.4.18 pwm_mc1_ch0_thre . . . . .	339
18.4.19 pwm_mc1_ch1_thre . . . . .	339
18.4.20 pwm_mc1_ch2_thre . . . . .	340
18.4.21 pwm_mc1_ch3_thre . . . . .	340
18.4.22 pwm_mc1_int_sts . . . . .	341
18.4.23 pwm_mc1_int_mask . . . . .	342
18.4.24 pwm_mc1_int_clear . . . . .	343
18.4.25 pwm_mc1_int_en . . . . .	343
<b>19 TIMER . . . . .</b>	<b>345</b>
<b>19.1 简介 . . . . .</b>	<b>345</b>
<b>19.2 主要特征 . . . . .</b>	<b>346</b>
<b>19.3 功能描述 . . . . .</b>	<b>346</b>
<b>19.3.1 通用定时器工作原理 . . . . .</b>	<b>347</b>
<b>19.3.2 看门狗定时器工作原理 . . . . .</b>	<b>349</b>
<b>19.3.3 报警设定 . . . . .</b>	<b>349</b>
<b>19.3.4 看门狗报警 . . . . .</b>	<b>349</b>
<b>19.4 寄存器描述 . . . . .</b>	<b>350</b>
<b>19.4.1 TCCR . . . . .</b>	<b>352</b>
<b>19.4.2 TMR2_0 . . . . .</b>	<b>352</b>
<b>19.4.3 TMR2_1 . . . . .</b>	<b>353</b>
<b>19.4.4 TMR2_2 . . . . .</b>	<b>353</b>
<b>19.4.5 TMR3_0 . . . . .</b>	<b>353</b>
<b>19.4.6 TMR3_1 . . . . .</b>	<b>354</b>
<b>19.4.7 TMR3_2 . . . . .</b>	<b>354</b>
<b>19.4.8 TCR2 . . . . .</b>	<b>354</b>
<b>19.4.9 TCR3 . . . . .</b>	<b>355</b>
<b>19.4.10 TSR2 . . . . .</b>	<b>355</b>
<b>19.4.11 TSR3 . . . . .</b>	<b>355</b>
<b>19.4.12 TIER2 . . . . .</b>	<b>356</b>
<b>19.4.13 TIER3 . . . . .</b>	<b>357</b>
<b>19.4.14 TPLVR2 . . . . .</b>	<b>357</b>
<b>19.4.15 TPLVR3 . . . . .</b>	<b>357</b>
<b>19.4.16 TPLCR2 . . . . .</b>	<b>358</b>
<b>19.4.17 TPLCR3 . . . . .</b>	<b>358</b>
<b>19.4.18 WMER . . . . .</b>	<b>359</b>
<b>19.4.19 WMR . . . . .</b>	<b>359</b>
<b>19.4.20 WVR . . . . .</b>	<b>360</b>
<b>19.4.21 WSR . . . . .</b>	<b>360</b>

19.4.22 TICR2 . . . . .	361
19.4.23 TICR3 . . . . .	361
19.4.24 WICR . . . . .	362
19.4.25 TCER . . . . .	362
19.4.26 TCMR . . . . .	363
19.4.27 TILR2 . . . . .	363
19.4.28 TILR3 . . . . .	364
19.4.29 WCR . . . . .	364
19.4.30 WFAR . . . . .	364
19.4.31 WSAR . . . . .	365
19.4.32 TCVWR2 . . . . .	365
19.4.33 TCVWR3 . . . . .	365
19.4.34 TCVSYN2 . . . . .	366
19.4.35 TCVSYN3 . . . . .	366
19.4.36 TCDR . . . . .	366
19.4.37 GPIO . . . . .	367
19.4.38 GPIO_LAT1 . . . . .	368
19.4.39 GPIO_LAT2 . . . . .	368
19.4.40 TCDR_FORCE . . . . .	368
20 I2S . . . . .	370
20.1 简介 . . . . .	370
20.2 主要特征 . . . . .	370
20.3 功能描述 . . . . .	370
20.4 功能描述 . . . . .	371
20.4.1 数据格式描述 . . . . .	371
20.4.2 基本架构图 . . . . .	373
20.4.3 时钟源 . . . . .	373
20.4.4 I2S 中断 . . . . .	374
20.5 寄存器描述 . . . . .	374
20.5.1 i2s_config . . . . .	375
20.5.2 i2s_int_sts . . . . .	376
20.5.3 i2s_bclk_config . . . . .	377
20.5.4 i2s_fifo_config_0 . . . . .	378
20.5.5 i2s_fifo_config_1 . . . . .	379
20.5.6 i2s_fifo_wdata . . . . .	379
20.5.7 i2s_fifo_rdata . . . . .	380
20.5.8 i2s_io_config . . . . .	380
21 PDM . . . . .	382
21.1 简介 . . . . .	382

21.2 主要特征 . . . . .	382
21.3 功能描述 . . . . .	382
21.3.1 PDM 中断 . . . . .	383
21.3.2 FIFO 格式控制 . . . . .	383
21.3.3 FIFO 的启动与 DMA 搬运 . . . . .	384
21.4 寄存器描述 . . . . .	384
21.4.1 audpdm_top . . . . .	385
21.4.2 audpdm_itf . . . . .	386
21.4.3 pdm_adc_0 . . . . .	387
21.4.4 pdm_adc_1 . . . . .	388
21.4.5 pdm_dac_0 . . . . .	389
21.4.6 pdm_pdm_0 . . . . .	391
21.4.7 pdm_rsvd0 . . . . .	391
21.4.8 pdm_dbg_0 . . . . .	392
21.4.9 pdm_dbg_1 . . . . .	392
21.4.10 pdm_dbg_2 . . . . .	393
21.4.11 pdm_dbg_3 . . . . .	393
21.4.12 pdm_dbg_4 . . . . .	394
21.4.13 pdm_adc_s0 . . . . .	395
21.4.14 pdm_adc_s1 . . . . .	395
21.4.15 pdm_adc_s2 . . . . .	395
21.4.16 pdm_rx_fifo_ctrl . . . . .	396
21.4.17 pdm_rx_fifo_status . . . . .	399
21.4.18 pdm_rx_fifo_data . . . . .	400
22 AUDIO . . . . .	402
22.1 简介 . . . . .	402
22.2 主要特征 . . . . .	402
22.3 功能描述 . . . . .	403
22.3.1 AUDIO 中断 . . . . .	403
22.3.2 FIFO 格式控制 . . . . .	404
22.3.3 FIFO 的启动与 DMA 搬运 . . . . .	405
23 PSRAM Contorller . . . . .	406
23.1 简介 . . . . .	406
23.2 主要特征 . . . . .	406
23.3 功能描述 . . . . .	406
23.4 寄存器描述 . . . . .	406
23.4.1 psram_configure . . . . .	407
24 Emac . . . . .	409
24.1 简介 . . . . .	409

24.2 主要特征 . . . . .	409
24.3 功能描述 . . . . .	410
24.4 时钟 . . . . .	411
24.5 收发缓冲描述符 (BD, Buffer Descriptor) . . . . .	411
24.6 PHY 交互 . . . . .	411
24.7 编程流程 . . . . .	412
24.7.1 PHY 初始化 . . . . .	412
24.7.2 发送数据帧 . . . . .	412
24.7.3 接收数据帧 . . . . .	413
24.8 寄存器描述 . . . . .	413
24.8.1 MODE . . . . .	414
24.8.2 INT_SOURCE . . . . .	416
24.8.3 INT_MASK . . . . .	417
24.8.4 IPGT . . . . .	418
24.8.5 PACKETLEN . . . . .	419
24.8.6 COLLCONFIG . . . . .	420
24.8.7 TX_BD_NUM . . . . .	420
24.8.8 MIIMODE . . . . .	421
24.8.9 MIICOMMAND . . . . .	422
24.8.10 MIIADDRESS . . . . .	422
24.8.11 MIITX_DATA . . . . .	423
24.8.12 MIIRX_DATA . . . . .	423
24.8.13 MIISTATUS . . . . .	424
24.8.14 MAC_ADDR0 . . . . .	424
24.8.15 MAC_ADDR1 . . . . .	425
24.8.16 HASH0_ADDR . . . . .	425
24.8.17 HASH1_ADDR . . . . .	425
24.8.18 TXCTRL . . . . .	426
25 USB . . . . .	427
25.1 简介 . . . . .	427
25.2 主要特征 . . . . .	427
25.3 功能描述 . . . . .	428
25.3.1 USB 使用步骤 . . . . .	428
25.3.2 部分寄存器配置及功能描述 . . . . .	428
25.3.3 USB 枚举阶段中断处理流程 . . . . .	428
26 ISO11898 . . . . .	429
26.1 简介 . . . . .	429
26.2 主要特征 . . . . .	429

26.3 功能介绍 . . . . .	429
26.3.1 发送缓冲区 (TXB) . . . . .	429
26.3.2 接收缓冲区 (RXB, RXFIFO) . . . . .	429
26.3.3 接收过滤器 (ACF) . . . . .	430
26.3.4 位流处理器 (BSP) . . . . .	430
26.3.5 位时序逻辑 (BTL) . . . . .	430
26.3.6 错误管理逻辑 (EML) . . . . .	430
26.4 功能描述 . . . . .	430
26.4.1 模式 . . . . .	430
26.4.1.1 自测模式 . . . . .	430
26.4.1.2 静默模式 . . . . .	430
26.4.1.3 复位模式 . . . . .	430
26.4.2 发送处理 . . . . .	432
26.4.2.1 发送流程 . . . . .	432
26.4.2.2 终止发送 . . . . .	432
26.4.2.3 自发自收 . . . . .	433
26.4.2.4 注意点 . . . . .	433
26.4.3 接收处理 . . . . .	433
26.4.3.1 接收流程 . . . . .	433
26.4.3.2 消息数量 . . . . .	433
26.4.3.3 接收缓冲区 . . . . .	433
26.4.4 标识符过滤 . . . . .	434
26.4.4.1 单滤波器配置 . . . . .	434
26.4.4.2 双滤波器配置 . . . . .	435
26.4.5 出错管理 . . . . .	438
26.4.5.1 仲裁失败 . . . . .	438
26.4.5.2 错误捕获 . . . . .	439
26.4.5.3 接收错误计数器 (RXERR) . . . . .	441
26.4.5.4 发送错误计数器 (TXERR) . . . . .	441
26.4.5.5 错误限值设定 . . . . .	441
26.4.6 位时序 . . . . .	442
26.4.6.1 波特率分频器 (BRP) . . . . .	442
26.4.6.2 同步跳转宽度 (SJW) . . . . .	442
26.4.6.3 采样 (SAM) . . . . .	443
26.4.6.4 时间段 (TSEG) . . . . .	443
27 MJPEG . . . . .	444
27.1 简介 . . . . .	444
27.2 主要特征 . . . . .	444

27.3 功能描述 . . . . .	445
27.3.1 输入配置 . . . . .	445
27.3.2 量化系数表 . . . . .	445
27.3.3 软件模式和连动模式 . . . . .	445
27.3.4 swap 模式 . . . . .	445
27.3.5 Kick 模式 . . . . .	446
27.3.6 jpg 功能 . . . . .	446
27.3.7 缓存图片信息 . . . . .	446
27.3.8 支持多种中断信息 (可独立开关配置) . . . . .	446
28 JDEC . . . . .	447
28.1 简介 . . . . .	447
28.2 主要特点 . . . . .	447
28.3 功能描述 . . . . .	447
28.3.1 输出配置 . . . . .	447
28.3.2 量化系数 . . . . .	448
28.3.3 swap 模式 . . . . .	448
28.3.4 skip 模式 . . . . .	448
28.3.5 缓冲区 . . . . .	448
28.3.6 操作流程 . . . . .	448
28.4 使用注意事项 . . . . .	449
28.5 寄存器描述 . . . . .	449
29 VENC . . . . .	450
29.1 简介 . . . . .	450
29.2 主要特点 . . . . .	450
29.3 功能描述 . . . . .	450
29.3.1 软件模式和连动模式 . . . . .	450
29.3.2 CBR/VBR 模式 . . . . .	451
29.3.3 双码流 . . . . .	451
29.3.4 ROI(Region of Interest) . . . . .	451
29.3.5 OSD(On-Screen Display) 编码区域 . . . . .	451
29.3.6 动态配置最大/最小量化参数 . . . . .	451
29.3.7 动态配置 I/P 帧目标位元 . . . . .	451
29.3.8 动态配置 I 帧距离 . . . . .	451
29.3.9 视频序列中断 . . . . .	451
29.3.10 帧中断 . . . . .	452
29.3.11 输入缓存溢出状态警示 . . . . .	452
30 NPU toolchain . . . . .	453
30.1 图形界面流程图 . . . . .	453
30.2 网络结构 . . . . .	453

30.3 ONNX 转换器 . . . . .	454
30.4 优化器 . . . . .	456
30.5 Generator & Simulator . . . . .	457
31 NPU . . . . .	458
31.1 简介 . . . . .	458
31.2 主要特点 . . . . .	458
31.3 功能列表 . . . . .	458
31.4 API 参考 . . . . .	459
31.5 数据结构参考 . . . . .	461
32 IPC . . . . .	467
32.1 简介 . . . . .	467
32.2 主要特征 . . . . .	467
32.3 功能描述 . . . . .	467
32.3.1 基本原理 . . . . .	467
32.3.2 操作流程 . . . . .	467
33 LowPower . . . . .	469
33.1 概述 . . . . .	469
33.2 主要特征 . . . . .	469
33.3 功能描述 . . . . .	470
33.3.1 电源域 . . . . .	470
33.3.2 唤醒源 . . . . .	471
33.3.3 功耗模式 . . . . .	472
33.3.4 IO 保持 . . . . .	473
33.4 寄存器描述 . . . . .	473
33.4.1 HBN_TIME_L . . . . .	474
33.4.2 HBN_TIME_H . . . . .	474
34 SEC ENG . . . . .	475
34.1 简介 . . . . .	475
34.1.1 AES 简介 . . . . .	475
34.1.2 MD5 简介 . . . . .	475
34.1.3 SHA 简介 . . . . .	475
34.1.4 CRC 简介 . . . . .	475
34.1.5 GMAC 简介 . . . . .	476
34.2 主要特征 . . . . .	476
34.3 原理描述 . . . . .	476
34.3.1 MD5 的实现: . . . . .	477
34.3.2 SHA256 的实现: . . . . .	477
34.3.3 GMAC 的原理 . . . . .	479

34.4 功能描述 . . . . .	480
34.4.1 AES 加速器 . . . . .	480
34.4.2 SHA 加速器 . . . . .	481
34.4.3 随机数发生器 . . . . .	482
34.4.4 GMAC(link 模式) . . . . .	482
34.5 寄存器描述 . . . . .	483
34.5.1 se_sha_0_ctrl . . . . .	485
34.5.2 se_sha_0_msa . . . . .	486
34.5.3 se_sha_0_status . . . . .	486
34.5.4 se_sha_0_endian . . . . .	487
34.5.5 se_sha_0_hash_l_0 . . . . .	487
34.5.6 se_sha_0_hash_l_1 . . . . .	488
34.5.7 se_sha_0_hash_l_2 . . . . .	488
34.5.8 se_sha_0_hash_l_3 . . . . .	488
34.5.9 se_sha_0_hash_l_4 . . . . .	489
34.5.10 se_sha_0_hash_l_5 . . . . .	489
34.5.11 se_sha_0_hash_l_6 . . . . .	489
34.5.12 se_sha_0_hash_l_7 . . . . .	490
34.5.13 se_sha_0_hash_h_0 . . . . .	490
34.5.14 se_sha_0_hash_h_1 . . . . .	490
34.5.15 se_sha_0_hash_h_2 . . . . .	491
34.5.16 se_sha_0_hash_h_3 . . . . .	491
34.5.17 se_sha_0_hash_h_4 . . . . .	491
34.5.18 se_sha_0_hash_h_5 . . . . .	492
34.5.19 se_sha_0_hash_h_6 . . . . .	492
34.5.20 se_sha_0_hash_h_7 . . . . .	492
34.5.21 se_sha_0_link . . . . .	493
34.5.22 se_sha_0_ctrl_prot . . . . .	493
34.5.23 se_aes_0_ctrl . . . . .	493
34.5.24 se_aes_0_msa . . . . .	494
34.5.25 se_aes_0_mda . . . . .	495
34.5.26 se_aes_0_status . . . . .	495
34.5.27 se_aes_0_iv_0 . . . . .	495
34.5.28 se_aes_0_iv_1 . . . . .	496
34.5.29 se_aes_0_iv_2 . . . . .	496
34.5.30 se_aes_0_iv_3 . . . . .	496
34.5.31 se_aes_0_key_0 . . . . .	497
34.5.32 se_aes_0_key_1 . . . . .	497
34.5.33 se_aes_0_key_2 . . . . .	497

34.5.34 se_aes_0_key_3 . . . . .	498
34.5.35 se_aes_0_key_4 . . . . .	498
34.5.36 se_aes_0_key_5 . . . . .	498
34.5.37 se_aes_0_key_6 . . . . .	499
34.5.38 se_aes_0_key_7 . . . . .	499
34.5.39 se_aes_0_key_sel . . . . .	499
34.5.40 se_aes_1_key_sel . . . . .	500
34.5.41 se_aes_0_endian . . . . .	500
34.5.42 se_aes_sboot . . . . .	501
34.5.43 se_aes_0_link . . . . .	501
34.5.44 se_aes_0_ctrl_prot . . . . .	502
34.5.45 se_trng_0_ctrl_0 . . . . .	502
34.5.46 se_trng_0_status . . . . .	503
34.5.47 se_trng_0_dout_0 . . . . .	503
34.5.48 se_trng_0_dout_1 . . . . .	504
34.5.49 se_trng_0_dout_2 . . . . .	504
34.5.50 se_trng_0_dout_3 . . . . .	504
34.5.51 se_trng_0_dout_4 . . . . .	505
34.5.52 se_trng_0_dout_5 . . . . .	505
34.5.53 se_trng_0_dout_6 . . . . .	505
34.5.54 se_trng_0_dout_7 . . . . .	506
34.5.55 se_trng_0_test . . . . .	506
34.5.56 se_trng_0_ctrl_1 . . . . .	507
34.5.57 se_trng_0_ctrl_2 . . . . .	507
34.5.58 se_trng_0_ctrl_3 . . . . .	507
34.5.59 se_trng_0_test_out_0 . . . . .	508
34.5.60 se_trng_0_test_out_1 . . . . .	508
34.5.61 se_trng_0_test_out_2 . . . . .	509
34.5.62 se_trng_0_test_out_3 . . . . .	509
34.5.63 se_trng_0_ctrl_prot . . . . .	509
34.5.64 se_pka_0_ctrl_0 . . . . .	510
34.5.65 se_pka_0_seed . . . . .	511
34.5.66 se_pka_0_ctrl_1 . . . . .	511
34.5.67 se_pka_0_rw . . . . .	512
34.5.68 se_pka_0_rw_burst . . . . .	512
34.5.69 se_pka_0_ctrl_prot . . . . .	513
34.5.70 se_cdet_0_ctrl_0 . . . . .	513
34.5.71 se_cdet_0_ctrl_1 . . . . .	514
34.5.72 se_cdet_0_ctrl_prot . . . . .	514

---

34.5.73 se_gmac_0_ctrl_0 . . . . .	515
34.5.74 se_gmac_0_lca . . . . .	515
34.5.75 se_gmac_0_status . . . . .	516
34.5.76 se_gmac_0_ctrl_prot . . . . .	516
34.5.77 se_ctrl_prot_rd . . . . .	517
35 版本信息 . . . . .	518

## 插图

1.1 系统框图 . . . . .	32
2.1 系统时钟架构 . . . . .	44
2.2 模块时钟架构 . . . . .	45
2.3 外设时钟架构 . . . . .	46
4.1 普通 GPIO 输出波形 . . . . .	51
4.2 默认电平为高电平, 逻辑 1 电平翻转输出波形 . . . . .	52
4.3 默认电平为低电平, 逻辑 0 电平翻转输出波形 . . . . .	52
4.4 默认电平为高电平, 逻辑 0 电平翻转, 逻辑 1 电平翻转输出波形 . . . . .	53
5.1 ADC 基本框图 . . . . .	57
5.2 ADC 时钟 . . . . .	59
6.1 DAC 基本框图 . . . . .	68
7.1 DMA 框图 . . . . .	73
7.2 外设类型选择 . . . . .	75
7.3 LLI 框架 . . . . .	77
8.1 工作原理 . . . . .	117
8.2 图像平移 . . . . .	118
8.3 图像旋转 . . . . .	119
8.4 图像翻折 . . . . .	120
8.5 图像填充 . . . . .	121
8.6 Color Key . . . . .	122
10.1 DBI 基本框图 . . . . .	143
10.2 写时序 . . . . .	144
10.3 读时序 . . . . .	145

10.4 RGB565 输出 . . . . .	146
10.5 RGB666 输出 . . . . .	147
10.6 RGB888 输出 . . . . .	148
10.7 写时序 . . . . .	149
10.8 读时序 . . . . .	150
10.9 RGB565 输出 . . . . .	150
10.10 RGB666 输出 . . . . .	151
10.11 RGB888 输出 . . . . .	151
10.12 写时序 . . . . .	152
10.13 读时序 . . . . .	153
10.14 RGB565 输出 . . . . .	153
10.15 RGB666 输出 . . . . .	154
10.16 RGB888 输出 . . . . .	154
10.17 输入像素格式 . . . . .	155
11.1 DPI 基本框图 . . . . .	165
11.2 DPI 时序图 . . . . .	166
11.3 定时参数图 . . . . .	167
11.4 定时参数取值范围表 . . . . .	168
11.5 测试模式 . . . . .	169
11.6 YUV422 处理过程 . . . . .	170
11.7 RGB888 处理过程 . . . . .	171
11.8 RGB565 处理过程 . . . . .	172
11.9 RGBA8888 处理过程 . . . . .	173
11.10 YUV420 处理过程 . . . . .	174
12.1 DSI 基本框图 . . . . .	177
12.2 协议分层 . . . . .	178
12.3 高速和低功耗模式下的电平 . . . . .	179
12.4 通道状态码 . . . . .	180
12.5 时钟通道模式切换图 . . . . .	181
12.6 由超低功耗模式进入低功耗模式 . . . . .	182
12.7 由高速时钟模式进入低功耗模式 . . . . .	182
12.8 由低功耗模式进入超低功耗模式 . . . . .	183
12.9 由低功耗模式进入高速时钟模式 . . . . .	183
12.10 数据通道三种模式进入和退出序列 . . . . .	184
12.11 高速数据传输进入序列 . . . . .	184
12.12 高速数据传输退出序列 . . . . .	185
12.13 高速数据传输突发序列 . . . . .	186
12.14 总线反转序列 . . . . .	186
12.15 Escape 模式序列 . . . . .	187
12.16 Escape 模式命令 . . . . .	188

12.17 低功耗数据传输序列 . . . . .	189
12.18 超低功耗状态序列 . . . . .	190
12.19 远程复位序列 . . . . .	191
12.20 应答序列 . . . . .	192
12.21 发送端拆分 . . . . .	193
12.22 接收端组合 . . . . .	194
12.23 非通道整数倍传输 . . . . .	195
12.24 短数据包结构图 . . . . .	196
12.25 长数据包结构图 . . . . .	196
12.26 一次传输多个数据包 . . . . .	197
12.27 数据包字节的位顺序 . . . . .	198
12.28 数据包的字节顺序 . . . . .	198
12.29 从 MCU 到显示模块的数据类型 . . . . .	200
12.30 从显示模块到 MCU 的数据类型 . . . . .	201
12.31 ECC 计算示例 . . . . .	202
12.32 CRC 计算方法 . . . . .	203
12.33 CRC 计算示例 . . . . .	203
12.34 具有同步脉冲的非突发模式时序图 . . . . .	204
12.35 具有同步事件的非突发模式时序图 . . . . .	205
12.36 突发模式时序图 . . . . .	206
12.37 YUV422 像素格式 . . . . .	207
12.38 RGB565 像素格式 . . . . .	208
12.39 RGB666 像素格式 . . . . .	209
12.40 RGB888 像素格式 . . . . .	210
13.1 Cam 框图 . . . . .	234
13.2 输入源选择 . . . . .	236
13.3 FIFO 框架 . . . . .	237
13.4 内存 . . . . .	238
14.1 NEC 逻辑波形 . . . . .	251
14.2 NEC 协议波形 . . . . .	251
14.3 RC5 逻辑波形 . . . . .	251
14.4 RC5 协议波形 . . . . .	252
15.1 SPI 时序图 . . . . .	265
15.2 SPI Ignore 波形图 . . . . .	266
15.3 SPI 滤波波形图 . . . . .	267
16.1 UART 数据格式 . . . . .	279
16.2 UART 架构图 . . . . .	279
16.3 UART 时钟 . . . . .	280
16.4 UART 采样波形图 . . . . .	281

16.5 UART 滤波波形图 . . . . .	282
16.6 UART 固定字符模式波形图 . . . . .	283
16.7 UART 硬件流控图 . . . . .	283
16.8 一个典型的 LIN 帧 . . . . .	284
16.9 LIN 的 Break 域 . . . . .	285
16.10 LIN 的 Sync 域 . . . . .	285
16.11 LIN 的 ID 域 . . . . .	285
17.1 I2C 起始和停止条件 . . . . .	305
17.2 I2C 数据传输格式 . . . . .	305
17.3 主发送和从接收的时序 . . . . .	306
17.4 主接收和从发送的时序 . . . . .	306
17.5 同时传输数据波形示意图 . . . . .	308
18.1 PWM 在不同配置下的波形图 . . . . .	322
19.1 定时器框图 . . . . .	345
19.2 看门狗定时器框图 . . . . .	346
19.3 定时器在 PreLoad 模式下工作时序 . . . . .	348
19.4 Watchdog 工作时序 . . . . .	349
19.5 看门狗报警机制 . . . . .	350
20.1 标准 I2S 数据格式 . . . . .	371
20.2 I2S 左对齐/右对齐数据格式 . . . . .	372
20.3 I2S TDM64 模式六通道录音 . . . . .	373
20.4 I2S 时钟 . . . . .	373
24.1 EMAC 框图 . . . . .	410
26.1 Single filter configuration, receiving standard frame messages . . . . .	434
26.2 Single filter configuration, receiving extended frame messages . . . . .	435
26.3 Dual filter configuration, receiving standard frame messages . . . . .	436
26.4 Dual filter configuration, receiving extended frame messages . . . . .	437
26.5 Arbitration lost bit number interpretation . . . . .	438
26.6 Example of arbitration lost bit number interpretation; result: ALC = 08 . . . . .	438
26.7 General structure of a bit period . . . . .	442
30.1 图形界面流程图 . . . . .	453
30.2 网络结构图 . . . . .	454
30.3 转换界面 . . . . .	455
30.4 显示或下载界面 . . . . .	456
30.5 优化器界面 . . . . .	456
30.6 预测选择界面 . . . . .	457

---

32.1 IPC 流程图 . . . . .	468
33.1 低功耗模式 . . . . .	469
34.1 AES 加密流程图 . . . . .	476
34.2 MD5 Padding . . . . .	477
34.3 消息认证码流程图 . . . . .	479
34.4 AES 运算模式图 . . . . .	480

## 表格

1.1 启动模式 . . . . .	37
1.2 地址映射 . . . . .	37
1.3 中断分配 . . . . .	39
1.4 中断分配 . . . . .	40
2.1 软件复位功能表 1 . . . . .	41
2.2 软件复位功能表 2 . . . . .	42
5.1 ADC 内部信号 . . . . .	58
5.2 ADC 外部引脚 . . . . .	58
5.3 ADC 转换结果含义 . . . . .	60
17.1 I2C 引脚 . . . . .	304
20.1 I2S 引脚 . . . . .	371
24.1 传输信号 . . . . .	411
26.1 不同模式下各寄存器含义说明 . . . . .	431
26.2 仲裁失败捕获的位置 . . . . .	438
26.3 错误捕获的类型 . . . . .	440
26.4 错误捕获的位置 . . . . .	440
33.1 电源模式 . . . . .	471
33.2 唤醒源 . . . . .	472
35.1 文档版本修改信息 . . . . .	518

## 系统和存储器概述

### 1.1 系统架构

BL808 系列芯片主要包含无线和多媒体两个子系统。

无线子系统包含一颗 RISC-V 32-bit 高性能 CPU, 集成 Wi-Fi BT/Zigbee 无线子系统, 可以实现多种无线连接和数据传输, 提供多样化的连接与传输体验。

多媒体子系统包含一颗 RISC-V 64-bit 超高性能 CPU, 集成 DVP/CSI/ H264/NPU 等视频处理模块, 可以广泛应用于视频监控/智能音箱等多种 AI 领域。

BL808 系统架构如下图所示:

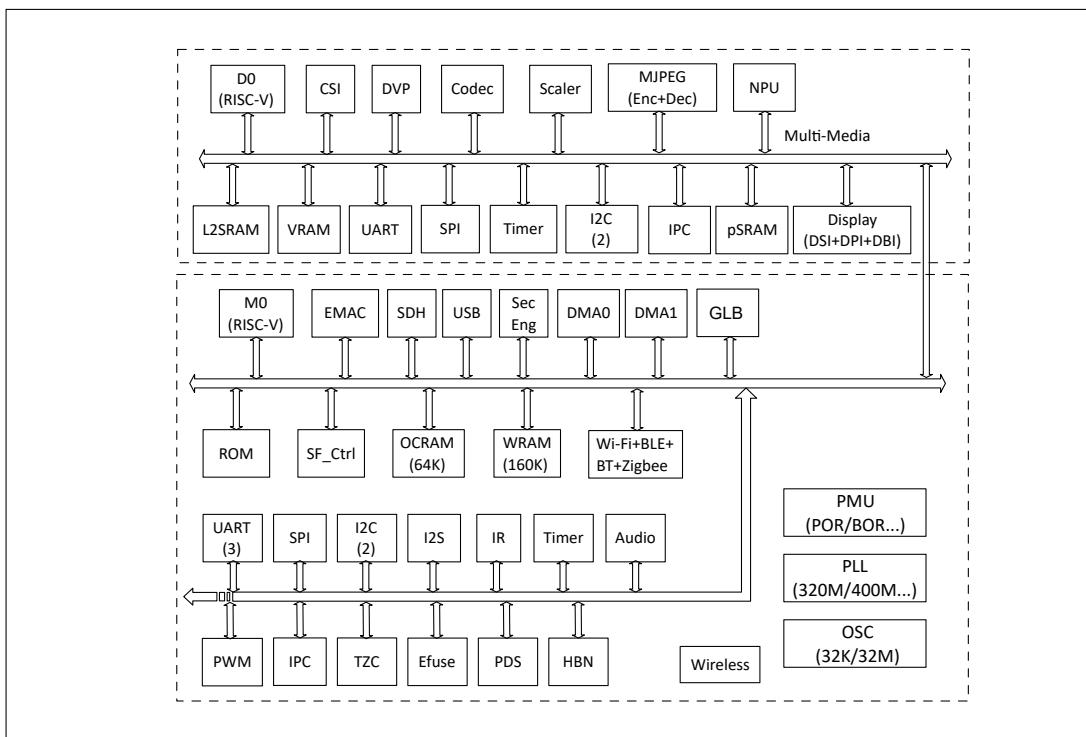


图 1.1: 系统框图

无线子系统有以下部分组成：

- 处理器
  - 集成了一颗 RISC-V 高性能 CPU(M0) 和一颗 RISC-V 低功耗 CPU(LP)
  - 支持多种无线连接和数据传输，提供多样化的连接与传输体验
- 总线接口：AXI 总线和 AHB 总线
  - CPU 主要通过 AXI 总线访问存储器
  - CPU 主要通过 AHB 总线访问外设
  - 具备低功耗，高性能等特性
- 外设
  - 包含 UART/SPI/I2C/PWM/SDH/EMAC/USB 等常见外设，支持与周边设备进行连接、控制和存储
  - 集成 Audio 模块，支持音频数据、模拟麦克风和数字麦克风的输入，方便用户开发音频相关应用
  - 适用于低功耗应用场景，例如：在智能音箱的低功耗应用场景时，关掉其它 CPU 和外设，LP 用于 VAD 的唤醒检测

多媒体子系统有以下部分组成：

- 处理器
  - 集成了一颗 RISC-V 超高性能 CPU(D0)
  - 适用于视频监控，智能门铃，智能猫眼，智能音箱等多种应用领域
  - NPU 模块用于神经网络算子的加速，适用于多种 AI 场景
- 外设
  - 包含 UART/SPI/I2C 等常规外设，方便系统功能开发
  - CSI 和 DVP 接口连接 CMOS Sensor，用于图像和视频的获取，内置 Codec 模块完成视频编码，配合 OSD，Scaler，2DDMA 等模块可以将视频数据流导入到 Display 模块显示

## 1.2 处理器

BL808 芯片内部包含 3 个 RISC-V 处理器，分别命名为 M0, D0, LP。M0 主要处理网络数据流和外设访问等功能，D0 主要功能是多媒体数据的计算和处理，配合多媒体相关外设，实现音视频数据的获取，存储和计算，LP 主要是负责低功耗模式下相关功能，实现系统低功耗。

## 1.2.1 M0 处理器介绍

### 1.2.1.1 主要功能特点

M0 是一颗 32-bit RISC-V CPU，它采用 5 级流水线结构：取指、译码、执行、内存访问、写回。是一款高性能嵌入式微处理处理器。其主要特征如下：

- 32 位 RISC 处理器；
- 支持 RISC-V RV32IMAFCP 指令集；
- 支持 RISC-V 32/16 位混编指令集；
- 支持 RISC-V 机器模式和用户模式；
- 32 个 32 位整型通用寄存器，32 个 32 位/64 位浮点通用寄存器；
- 整型 5 级/浮点 7 级，单发射，顺序执行流水线；
- 支持 AXI4.0 主设备接口以及 AHB5.0 外设接口；
- 32K 指令 cache，两路组相连结构；
- 16K 数据 cache，两路组相连结构；
- 支持非对齐内存访问；
- 双周期硬件乘法器，基 4 硬件除法器；
- 支持 BHT(8K) 和 BTB；
- 支持扩展增强指令集；
- 支持 MCU 特性扩展技术，包括中断处理加速技术、MCU 扩展特性；
- 兼容 RISC-V CLIC 中断标准，支持中断嵌套，外部中断源数量 96 个，有 4 个 bits 可以用于配置中断优先级；
- 兼容 RISC-V PMP 内存保护标准，8 个区域可配置；
- 支持性能监测单元 (HPM)；
- 支持 RISC-V Debug 协议标准；

### 1.2.1.2 扩展功能

- 扩展实现了位操作指令，算术运算指令，内存访问增强指令
- 扩展实现了 CACHE 操作指令，同步指令等方便软件人员编程
- 扩展实现了中断投机压栈技术和矢量中断咬尾技术，加速中断响应，中断响应延时为 20 个处理器时钟周期
- 扩展实现了如 NMI，深浅睡眠低功耗模式，低功耗唤醒事件，锁定等 MCU 领域常见的应用需求
- 支持非对齐内存访问，并可软件开关，方便软件人员编程和调试

### 1.2.1.3 实现标准

M0 兼容 RISC-V 标准，具体版本为：

- The RISC-V Instruction Set Manual, Volume I: RISC-V User-Level ISA, Version 2.2
- The RISC-V Instruction Set Manual, Volume II: RISC-V Privileged Architecture, Version1.10
- RISC-V Core-Local Interrupt Controller (CLIC) Version 0.8
- RISC-V External Debug Support Version 0.13.2
- RISC-V “P” Extension Proposal Version 0.9

## 1.2.2 D0 处理器介绍

### 1.2.2.1 主要功能特点

D0 是一颗 64-bit RISC-V CPU，它采用 5 级流水线结构。是一款超高性能嵌入式微处理处理器。可用于安防监控，智能语音处理，智能视频处理等领域。其主要特征如下：

- 64 位 RISC 处理器；
- 支持 RISC-V RV64IMAFCV 指令架构；
- 5 级单发按序执行流水线；
- 一级哈佛结构的指令和数据缓存，大小均为 32KB，缓存行为 64B；
- Sv39 内存管理单元，实现虚实地址转换与内存管理；
- jTLB 支持 128 个 entry；
- 支持 AXI4.0 128 比特 Master 接口；
- 支持核内中断 CLINT 和中断控制器 PLIC；
- 外部中断源数量 80 个，有 3 个 Bits 可以用于配置中断优先级；

- 支持 BHT (8K) 和 BTB;
- 兼容 RISC-V PMP 内存保护标准，8 个区域可配置；
- 支持性能监测单元 (HPM);

矢量计算单元的主要特征点如下：

- 遵循 RISC-V V 矢量扩展标准 (revision 0.7.1);
- 支持配置矢量执行单元 (128bits);
- 支持 INT8/INT16/INT32/FP16/FP32 矢量运算；
- 支持 segment load、store 指令；

### 1.2.2.2 扩展功能

- 扩展实现了位操作指令，算术运算指令，内存访问增强指令
- 扩展实现了 CACHE 操作指令，同步指令等方便软件人员编程
- 支持非对齐内存访问，并可软件开关，方便软件人员编程和调试

### 1.2.2.3 实现标准

D0 兼容 RISC-V 标准，具体版本为：

- The RISC-V Instruction Set Manual, Volume I: RISC-V User-Level ISA, Version 2.2
- The RISC-V Instruction Set Manual, Volume II: RISC-V Privileged Architecture, Version1.10
- RISC-V “V” Vector Extension, Version 0.7.1-20190610-Workshop-Release.
- RISC-V External Debug Support Version 0.13.2

## 1.2.3 LP 处理器介绍

### 1.2.3.1 主要功能特点

LP 是一颗 32-bit RISC-V CPU，它采用 2 级流水线结构，即指令提取级，指令译码、执行与回写级。它具有极低功耗、极低成本、高代码密度等特点，其主要特征如下：

- 32 位 RISC 处理器；
- 支持 RISC-V RV32EMC 指令集；
- 支持 RISC-V 32/16 位混编指令集；

- 16 个 32 位整型通用寄存器;
- 两级顺序执行流水线;
- 支持 RISC-V 机器模式和用户模式;
- 兼容 RISC-V CLIC 中断标准，支持中断嵌套，外部中断源 32 个。
- 支持 AHB-Lite 总线协议，支持指令总线，系统总线;
- 无内部 cache
- 无 PMP
- 支持 2 线调试接口

### 1.2.3.2 实现标准

LP 兼容 RISC-V 标准，具体版本为：

- The RISC-V Instruction Set Manual, Volume I: RISC-V User-Level ISA, Version 2.2
- The RISC-V Instruction Set Manual, Volume II: RISC-V Privileged Architecture, Version1.10
- RISC-V Core-Local Interrupt Controller (CLIC) Version 0.8

## 1.3 启动

系统支持从 Flash/UART/USB 启动，各个启动模式说明如下：

表 1.1: 启动模式

启动引脚	电平	描述
GPIO39	1	从 UART(GPIO20/21)/USB 启动，该模式主要用于 Flash 下载或者下载镜像到 RAM 执行
	0	从 Flash 启动应用镜像

## 1.4 内存

### 1.5 地址映射

表 1.2: 地址映射

模块	目标	开始地址	大小	描述
FLASH	FlashA	0x58000000	64MB	应用程序地址空间

表 1.2: 地址映射 (continued)

模块	目标	开始地址	大小	描述
pSRAM	pSRAM	0x50000000	64MB	pSRAM 存储器地址空间, 实际大小取决于芯片型号
RAM	OCRAM(MCU)	0x22020000	64KB	On Chip RAM 地址空间, 主要是供 M0 应用程序数据使用的内存
	WRAM(MCU)	0x22030000	160KB	Wireless RAM 地址空间, 主要是供 M0 无线网络数据使用的内存
	XRAM(EMI)	0x40000000	16KB	Shared RAM, 主要用于 M0/LP/D0 等各个 CPU 之间数据通信
	DRAM(MM)	0x3EF80000	512KB	Multimedia 側的 RAM 地址空间, 主要是供 D0 应用程序数据/H264/NPU 等模块使用的内存
	VRAM(MM)	0x3F000000	32KB	Multimedia 側的 RAM 地址空间, 主要是供 D0 应用程序数据/H264/NPU 等模块使用的内存
MMPERI	TIMER1	0x30009000	4KB	TIMER1 控制寄存器
	SPI1	0x30008000	4KB	SPI1 控制寄存器
	MM_GLB	0x30007000	4KB	Multimedia 側全局寄存器
	DMA2D	0x30006000	4KB	DMA2D 控制寄存器
	I2C3	0x30004000	4KB	I2C3 控制寄存器
	I2C2	0x30003000	4KB	I2C2 控制寄存器
	UART3	0x30002000	4KB	UART3 控制寄存器
	DMA2	0x30001000	4KB	DMA2 控制寄存器
MCUPERI	DMA1	0x20071000	4KB	DMA1 控制寄存器
	EMAC	0x20070000	4KB	EMAC 控制寄存器
	AUDIO	0x20055000	4KB	Audio 控制寄存器
	USB	0x20072000	4KB	USB 控制寄存器
	HBN	0x2000F000	4KB	深度睡眠控制 (休眠) 寄存器
	PDS	0x2000E000	4KB	睡眠控制 (掉电睡眠) 寄存器
	DMA0	0x2000C000	4KB	DMA0 控制寄存器
	I2S	0x2000AB00	256B	I2S 控制寄存器
	CAN_FD	0x2000AA00	256B	CAN 总线控制寄存器
		0x2000AA00	256B	UART2 控制寄存器
	I2C1	0x2000A900	256B	I2C1 控制寄存器
	IR	0x2000A600	256B	IR 控制寄存器
	TIMER0	0x2000A500	256B	TIMER0 控制寄存器
	PWM	0x2000A400	256B	PWM 控制寄存器
	I2C0	0x2000A300	256B	I2C0 控制寄存器
	SPI0	0x2000A200	256B	SPI0 控制寄存器
	UART1	0x2000A100	256B	UART1 控制寄存器
	UART0	0x2000A000	256B	UART0 控制寄存器
	eFuse	0x20056000	4KB	eFuse 存储器控制寄存器
	TZ	0x20005000	4KB	TrustZone 控制寄存器
	SEC_ENG	0x20004000	4KB	安全引擎控制寄存器

表 1.2: 地址映射 (continued)

模块	目标	开始地址	大小	描述
MCUPERI	GPIP	0x20002000	1KB	通用 DAC / ADC / ACOMP 接口控制寄存器
	GLB	0x20000000	4KB	全局控制寄存器
ROM	ROM	0x90000000	128KB	Bootrom 区域地址空间

## 1.6 中断源

CPU\_M0 和 CPU\_LP 一共包含 23 个中断源，中断源与对应的中断号如下表所示：

表 1.3: 中断分配

中断源		中断号	描述
DMA	DMA0_ALL	IRQ_NUM_BASE+15	DMA0 ALL Interrupt
	DMA1_ALL	IRQ_NUM_BASE+16	DMA1 ALL Interrupt
IR	IRTX	IRQ_NUM_BASE+19	IR TX Interrupt
	IRRX	IRQ_NUM_BASE+20	IR RX Interrupt
USB	USB	IRQ_NUM_BASE+21	USB Interrupt
EMAC	EMAC	IRQ_NUM_BASE+24	EMAC Interrupt
ADC	GPADC_DMA	IRQ_NUM_BASE+25	GPADC_DMA Interrupt
SPI	SPI0	IRQ_NUM_BASE+27	SPI Interrupt
UART	UART0	IRQ_NUM_BASE+28	UART0 Interrupt
	UART1	IRQ_NUM_BASE+29	UART1 Interrupt
	UART2	IRQ_NUM_BASE+30	UART2 Interrupt
GPIO	GPIO_DMA	IRQ_NUM_BASE+31	GPIO DMA Interrupt
I2C	I2C0	IRQ_NUM_BASE+32	I2C0 Interrupt
	I2C1	IRQ_NUM_BASE+39	I2C1 Interrupt
PWM	PWM	IRQ_NUM_BASE+33	PWM Interrupt
TIMER0	TIMER0_CH0	IRQ_NUM_BASE+36	Timer0 Channel 0 Interrupt
	TIMER0_CH1	IRQ_NUM_BASE+37	Timer0 Channel 1 Interrupt
	TIMER0_WDT	IRQ_NUM_BASE+38	Timer0 Watch Dog Interrupt
I2S	I2S	IRQ_NUM_BASE+40	I2S Interrupt
GPIO	GPIO_INT0	IRQ_NUM_BASE+44	GPIO Interrupt
PDS	PDS_WAKEUP	IRQ_NUM_BASE+50	PDS Wakeup Interrupt
HBN	HBN_OUT0	IRQ_NUM_BASE+51	Hibernate out 0 Interrupt
	HBN_OUT1	IRQ_NUM_BASE+52	Hibernate out 1 Interrupt

CPU\_D0 一共包含 21 个中断源，中断源与对应的中断号如下表所示：

表 1.4: 中断分配

中断源		中断号	描述
UART	UART3	IRQ_NUM_BASE+4	UART3 Interrupt
I2C	I2C2	IRQ_NUM_BASE+5	I2C2 Interrupt
	I2C3	IRQ_NUM_BASE+6	I2C3 Interrupt
SPI	SPI1	IRQ_NUM_BASE+7	SPI1 Interrupt
DMA2	DMA2_INT0	IRQ_NUM_BASE+24	DMA INT0 Interrupt
	DMA2_INT1	IRQ_NUM_BASE+25	DMA INT1 Interrupt
	DMA2_INT2	IRQ_NUM_BASE+26	DMA INT2 Interrupt
	DMA2_INT3	IRQ_NUM_BASE+27	DMA INT3 Interrupt
	DMA2_INT4	IRQ_NUM_BASE+28	DMA INT4 Interrupt
	DMA2_INT5	IRQ_NUM_BASE+29	DMA INT5 Interrupt
	DMA2_INT6	IRQ_NUM_BASE+30	DMA INT6 Interrupt
	DMA2_INT7	IRQ_NUM_BASE+31	DMA INT7 Interrupt
EMAC	EMAC2	IRQ_NUM_BASE+36	EMAC2 Interrupt
DMA2D	DMA2D_INT0	IRQ_NUM_BASE+45	DMA2D INT0 Interrupt
	DMA2D_INT1	IRQ_NUM_BASE+46	DMA2D INT1 Interrupt
PWM	PWM	IRQ_NUM_BASE+48	PWM1 Interrupt
TIMER1	TIMER0_CH0	IRQ_NUM_BASE+61	Timer1 Channel 0 Interrupt
	TIMER0_CH1	IRQ_NUM_BASE+62	Timer1 Channel 1 Interrupt
	TIMER0_WDT	IRQ_NUM_BASE+63	Timer1 Watch Dog Interrupt
AUDIO	AUDIO	IRQ_NUM_BASE+64	Audio Interrupt
PDS	PDS	IRQ_NUM_BASE+66	PDS Interrupt

注解：其中 IRQ\_NUM\_BASE 为 16，中断号 0-15 为 RISC-V 保留中断。

## 复位和时钟

### 2.1 简介

芯片内的时钟源：XTAL，PLL，RC。搭配分频等配置送至各模块。

### 2.2 复位管理

表 2.1: 软件复位功能表 1

BL808	RST_PIN /Watch Dog / PDS Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS	PDS /CPU	PDS
CPU(M0)	✓			✓	
CPU(LP)	✓				
bus	✓		✓		
glb	✓	swrst_s1[0]			
mix	✓	swrst_s1[1]			✓
gpip	✓	swrst_s1[2]	✓		
sec_eng	✓	swrst_s1[4]	✓		
TZ	✓		✓		
efuse	✓				
dma	✓	swrst_s1[12]	✓		
psram	✓	swrst_s1_ext[2]	✓		
usb	✓	swrst_s1_ext[3]	✓		
emac	✓	swrst_s1_ext[3]	✓		
audio	✓	swrst_s1_ext[5]	✓		
dma2	✓	swrst_s1_ext[8]	✓		
pds		swrst_s1[14]			

表 2.1: 软件复位功能表 1(continued)

BL808	RST_PIN /Watch Dog / PDS Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS	PDS /CPU	PDS
uart0	✓	swrst_s1a[0]	✓		
uart1	✓	swrst_s1a[1]	✓		
spi	✓	swrst_s1a[2]	✓		
i2c	✓	swrst_s1a[3]	✓		
pwm	✓	swrst_s1a[4]	✓		
timer	✓	swrst_s1a[5]	✓		
irr	✓	swrst_s1a[6]	✓		
uart2/can	✓	swrst_s1a[10]	✓		
i2s	✓	swrst_s1a[11]	✓		
pdm	✓	swrst_s1a[12]	✓		
wifi	✓	swrst_s2[0]			✓
ble	✓	swrst_s3[0][2]			✓

表 2.2: 软件复位功能表 2

BL808_MM	Watch Dog / Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])	MMCPU0
CPU(D0)	✓			✓
bus	✓		✓	
mm_misc	✓	swrst_mm_misc	✓	
mm_dma	✓	swrst_dma	✓	
mm_2ddma	✓	swrst_dma2d	✓	
mm_uart	✓	swrst_uart0	✓	
mm_i2c	✓	swrst_i2c0	✓	
mm_ipc	✓	swrst_i2c1	✓	
mm_timer	✓	swrst_timer	✓	
uhs_ctrl	✓	swrst_pUHS	✓	
disp_tsrc	✓	swrst_dp_tsrc	✓	
nr3d_ctrl	✓	swrst_nr3d_ctrl	✓	
dvp2busA	✓	swrst_dvp2busA	✓	
dvp2busB	✓	swrst_dvp2busB	✓	
dvp2busC	✓	swrst_dvp2busC	✓	

表 2.2: 软件复位功能表 2(continued)

BL808_MM	Watch Dog / Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])	MMCPU0
dvp2busD	✓	swrst_dvp2busD	✓	
dvp2busE	✓	swrst_dvp2busE	✓	
dvp2busF	✓	swrst_dvp2busF	✓	
dvp2busG	✓	swrst_dvp2busG	✓	
dvp2busH	✓	swrst_dvp2busH	✓	
jdec	✓	swrst_mjpeg_dec	✓	
blai	✓	swrst_cnn	✓	

## 2.3 时钟源

时钟源包含：

- XTAL：外部晶振时钟，视系统需求频率可选 24、32、38.4、40MHz
- XTAL32K：外部晶振时钟，频率 32kHz
- RC32K：RC 振荡器时钟，频率 32kHz，提供校准
- RC32M：RC 振荡器时钟，频率 32MHz，提供校准
- PLL：多个 PLL 模块，可以产生若干不同频率的时钟，以满足各个应用场景

时钟控制单元将来自振荡器的时钟分配给内核和外围设备。可通过选择系统时钟源，动态分频器，时钟配置，睡眠使用 32kHz 时钟，以达到低功耗时钟管理。

外围设备时钟包括: Flash、UART、I2C、SPI、PWM、IR-remote、ADC、DAC。

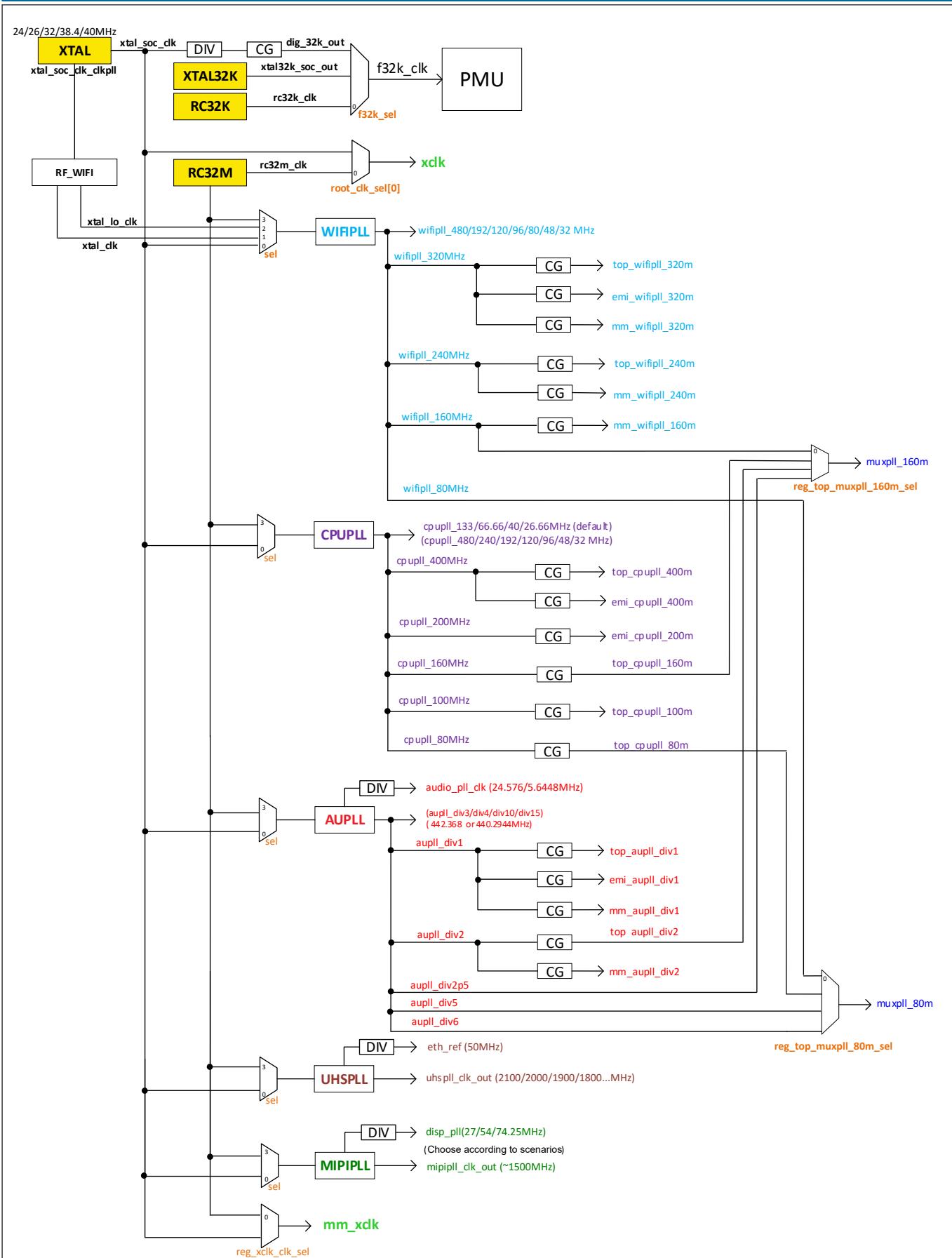


图 2.1: 系统时钟架构

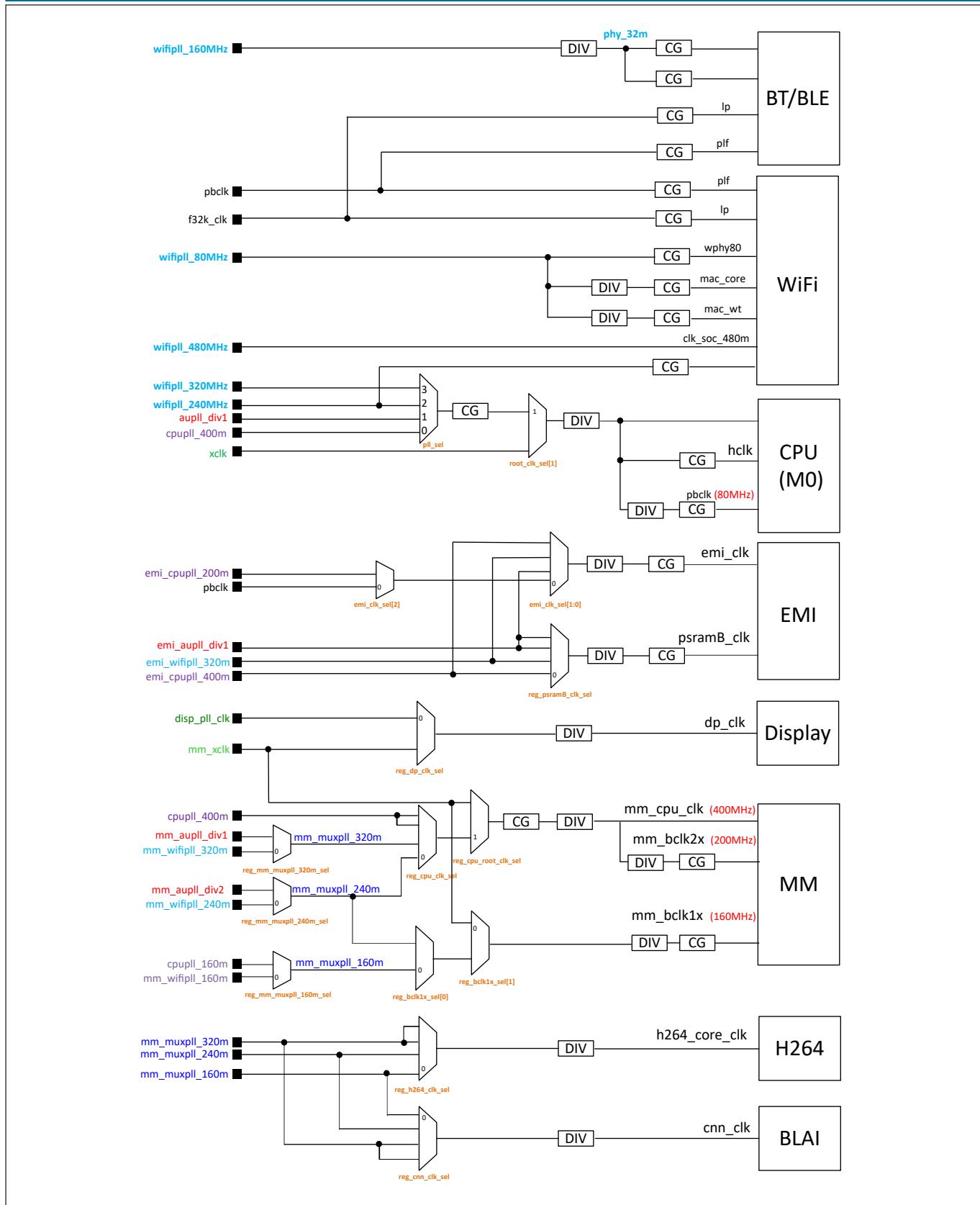


图 2.2: 模块时钟架构

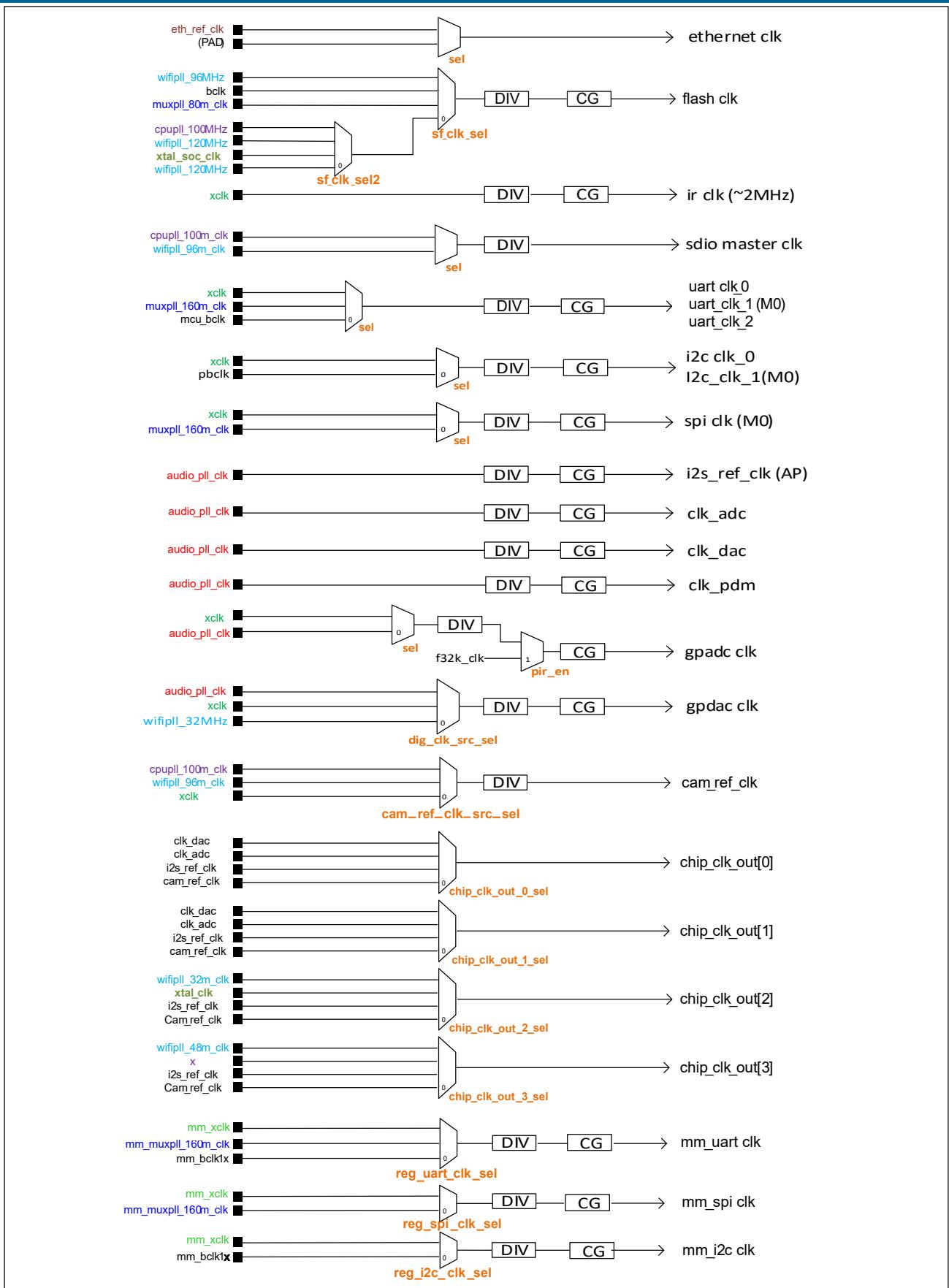


图 2.3: 外设时钟架构

## 3.1 简介

## 3.2 功能描述

### 3.2.1 时钟管理

时钟管理功能主要用于设定处理器、总线、各个外设的时钟，通过该模块可以设定上述模块工作的时钟源，时钟分频等，同时也可以实现对上述模块时钟的门控，以达到系统低功耗的目的。详细设定可以参考系统时钟相关的章节。

### 3.2.2 复位管理

复位管理提供各个外设模块的单独复位功能以及芯片复位功能。

芯片复位包括：

- CPU 复位：仅仅复位 CPU 模块，程序会重新运行，外设不会被复位
- 系统复位：各个外设和 CPU 会被复位，但是 AON 域的相关寄存器不会被复位
- 上电复位：整个系统包括 AON 域的相关寄存器都会被复位

应用程序可以根据需要，选择使用对应的复位方式。

### 3.2.3 总线管理

提供总线的仲裁设定以及总线出错设定，可以设定在总线出错时候是否产生中断，并提供出错总线地址信息，方便用户调试程序。

### 3.2.4 内存管理

提供各个内存模块在芯片系统低功耗模式时的功耗管理，包含两种设定模式：

- **retention** 模式：在该模式下，内存上的数据可以保存，但是在退出低功耗模式之前，无法读写。
- **sleep** 模式：在该模式下，内存的数据会丢失，仅用于降低系统功耗。

## 4.1 简介

GPIO(General Purpose I/O Ports) 是通用输入/输出端口，用户可将其与外部硬件设备连接达到控制外部硬件设备的目的。

## 4.2 主要特征

- 最多可达 40 个 I/O 引脚
- 每个 I/O 引脚可配置的功能最多可达 25 种
- 每个 I/O 引脚都可以配置为上拉、下拉或浮空模式
- 每个 I/O 引脚都可以配置为输入、输出或高阻态模式
- 每个 I/O 引脚的输出模式都有 4 种驱动能力可选择
- 每个 I/O 引脚的输入模式都可以设置是否开启施密特触发器
- 每个 I/O 引脚都支持 9 种外部中断模式
- FIFO 深度为 128 半字
- 可用 DMA 将数据从 RAM 搬至 I/O 引脚用于输出

## 4.3 GPIO 输入设置

通过设置 GPIO\_CFG<sub>xx</sub> 寄存器，将通用的接口配置为输入模式，过程如下（其中 **xx** 表示 GPIO 引脚号）：

- 将 `<reg_gpio_xx_ie>` 设置为 1，使能 GPIO 输入模式
- 将 `<reg_gpio_xx_func_sel>` 设置为 11 即进入 SW-GPIO 模式
- 在 SW-GPIO 模式下，可通过 `<reg_gpio_xx_smt>` 设置是否使能施密特触发器，用于波形的整形

- 通过 `<reg_gpio_xx_pu>` 和 `<reg_gpio_xx_pd>` 设置是否使能内部上拉和下拉功能
- 如果需要中断则通过 `<reg_gpio_xx_int_mode_set>` 设定外部中断的类型，此时即可通过 `<reg_gpio_xx_i>` 读取 I/O 引脚的电平值

## 4.4 GPIO 输出设置

通过 `GPIO_CFGxx` 寄存器设置 GPIO 不同的输出模式，共支持以下四种模式。

### 4.4.1 普通输出模式

- 将 `<reg_gpio_xx_oe>` 设置为 1，使能 GPIO 输出模式
- 将 `<reg_gpio_xx_func_sel>` 设置为 11，进入 SW-GPIO 模式
- 将 `<reg_gpio_xx_mode>` 设置为 0，使能 I/O 的普通输出功能
- 可通过 `<reg_gpio_xx_pu>` 和 `<reg_gpio_xx_pd>` 设置是否使能内部上拉和下拉功能，此时即可通过 `<reg_gpio_xx_o>` 设置 I/O 引脚的电平值。

### 4.4.2 Set/Clear 输出模式

- 将 `<reg_gpio_xx_oe>` 设置为 1，使能 GPIO 输出模式
- 将 `<reg_gpio_xx_func_sel>` 设置为 11，进入 SW-GPIO 模式
- 将 `<reg_gpio_xx_mode>` 设置为 1，使能 I/O 的 Set/Clear 输出功能
- 可通过 `<reg_gpio_xx_pu>` 和 `<reg_gpio_xx_pd>` 设置是否使能内部上拉和下拉功能

在 Set/Clear 输出模式下，可将 `<reg_gpio_xx_set>` 置 1，使对应 I/O 引脚设为高电平，将 `<reg_gpio_xx_clr>` 置 1 可将对应 I/O 引脚设为低电平，如果同时将 `<reg_gpio_xx_set>` 和 `<reg_gpio_xx_clr>` 置 1 则对应 I/O 引脚为高电平，对 `<reg_gpio_xx_set>` 和 `<reg_gpio_xx_clr>` 写 0 没有作用。

### 4.4.3 可编程输出模式

- 将 `<reg_gpio_xx_oe>` 设置为 1，使能 GPIO 输出模式
- 将 `<reg_gpio_xx_func_sel>` 设置为 11，进入 SW-GPIO 模式
- 将 `<reg_gpio_xx_mode>` 设置为 2，使能 I/O 的可编程输出功能
- 可通过 `<reg_gpio_xx_pu>` 和 `<reg_gpio_xx_pd>` 设置是否使能内部上拉和下拉功能。

在可编程输出模式下，当 `GPIO_CFG142` 寄存器的位 `<cr_gpio_tx_en>` 置 1 时，`GPIO_CFG144` 寄存器存储的数据，依照顺序逐个写入对应的 I/O 引脚，从而设置引脚的电平，缓冲区大小为  $128 * 16\text{bits}$ 。需要注意的是 `GPIO_CFG144` 寄存器储存的数据不是实际的引脚电平值，而是一种可编程的逻辑 0/1，逻辑 0/1 与实际电平之间的关系描述如下。

在此模式下输出到引脚的电平状态可以自由设置。以 `XCLK` 为时钟源，`GPIO_CFG142` 寄存器中 `<cr_code_total_time>`

设置的数值为一个周期：

逻辑 1 的电平状态：由 `<cr_code1_high_time>` 设置的一段高电平加上 `<cr_code_total_time>-<cr_code1_high_time>` 设置的一段低电平，当 `<cr_invert_code1_high>` 为 0 时，表示逻辑 1 先输出高电平再输出低电平，否则先输出低电平再输出高电平；

逻辑 0 的电平状态：由 `<cr_code0_high_time>` 设置一段高电平加上 `<cr_code_total_time>-<cr_code0_high_time>` 设置一段低电平，当 `<cr_invert_code0_high>` 为 0 时，表示逻辑 0 先输出高电平再输出低电平，否则先输出低电平再输出高电平。

---

注解：由于缓冲区的寄存器是 16 位宽，所以每 16 个引脚为一组，在一组中从低到高序号的引脚分别由缓冲区中对应的 bit 控制。此外，GPIO\_CFG143 寄存器中的位 `<cr_gpio_dma_out_sel_latch>` 应该设置为 0。`<cr_gpio_dma_park_value>` 用于设置 I/O 默认电平，为 1 时默认电平是高电平，为 0 时默认电平是低电平。

---

当 `<cr_code_total_time> = 10`, `<cr_code0_high_time> = 1`, `<cr_code1_high_time> = 5`, `<cr_invert_code0_high> = 0`, `<cr_invert_code1_high> = 0`, `<cr_gpio_dma_park_value> = 0`, `<cr_gpio_dma_out_sel_latch> = 0` 时，对应的波形图如下所示：

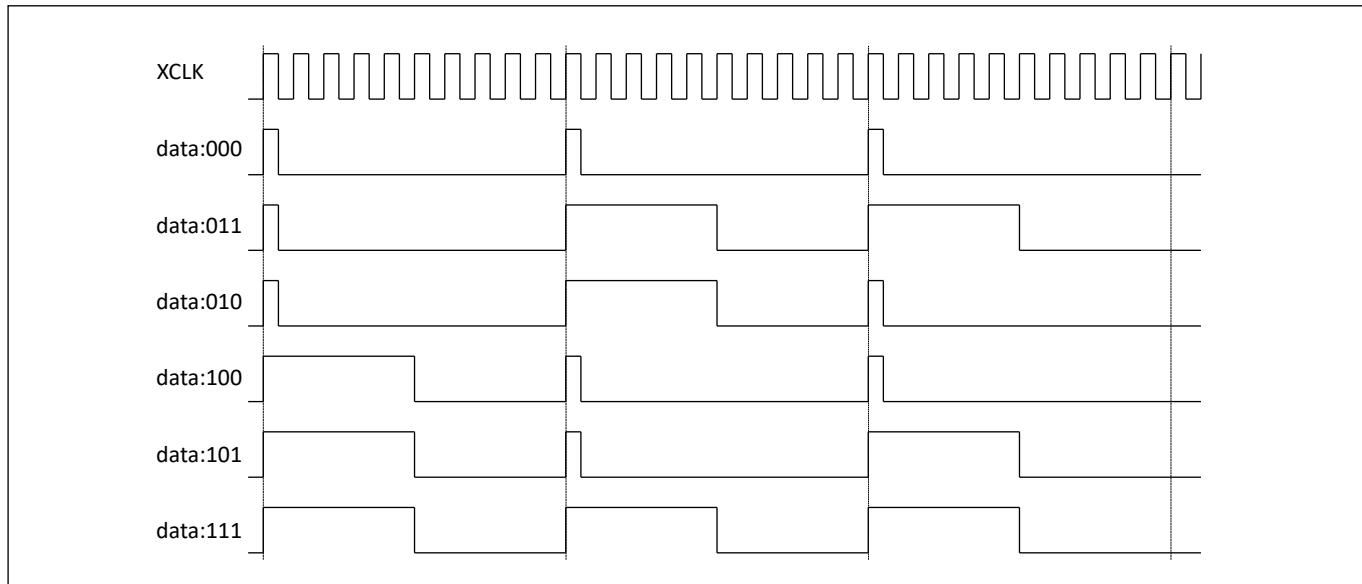


图 4.1: 普通 GPIO 输出波形

当 `<cr_code_total_time> = 10`, `<cr_code0_high_time> = 1`, `<cr_code1_high_time> = 5`, `<cr_invert_code0_high> = 0`, `<cr_invert_code1_high> = 1`, `<cr_gpio_dma_park_value> = 1`, `<cr_gpio_dma_out_sel_latch> = 0` 时，对应的波形图如下所示：

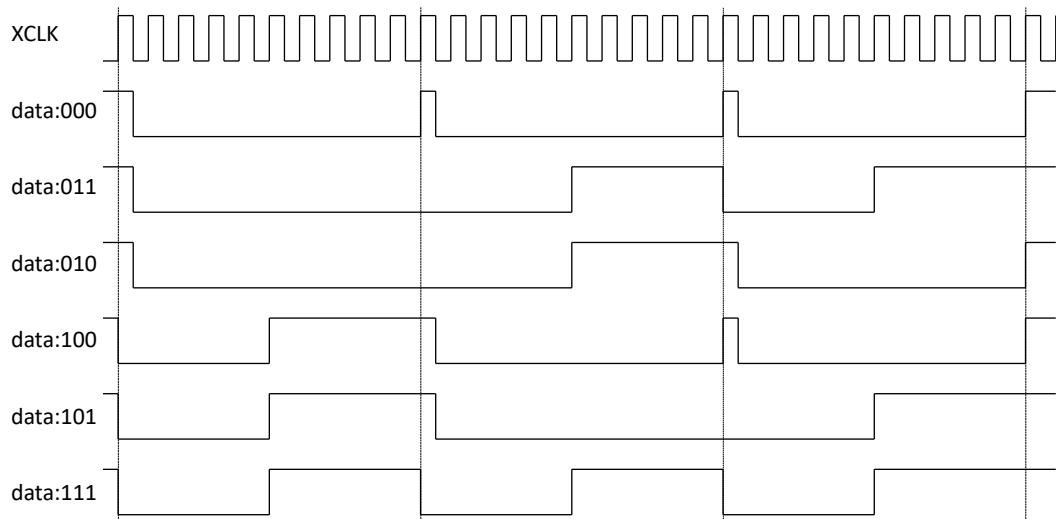
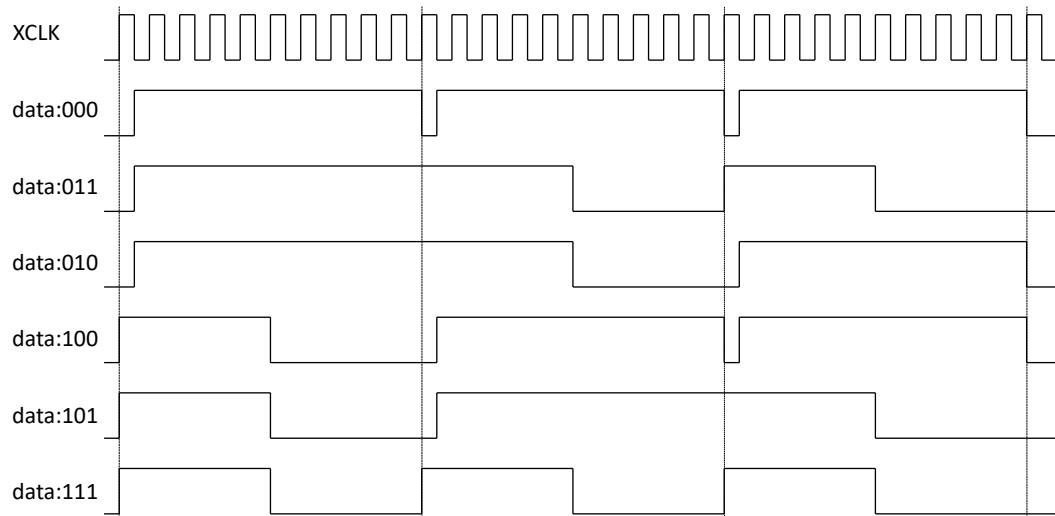


图 4.2: 默认电平为高电平, 逻辑 1 电平翻转输出波形

当 `<cr_code_total_time>` = 10, `<cr_code0_high_time>` = 1, `<cr_code1_high_time>` = 5, `<cr_invert_code0_high>` = 1, `<cr_invert_code1_high>` = 0, `<cr_gpio_dma_park_value>` = 0, `<cr_gpio_dma_out_sel_latch>` = 0 时, 对应的波形图如下所示:



configuration: T12 = 10, L0T1 = 1, L1T1 = 5, L0S1 = 1, L1S1 = 0, PS = 0, TM = 0

Example 3 of GPIO output tx configuration

图 4.3: 默认电平为低电平, 逻辑 0 电平翻转输出波形

当 `<cr_code_total_time>` = 10, `<cr_code0_high_time>` = 1, `<cr_code1_high_time>` = 5, `<cr_invert_code0_high>` = 1, `<cr_invert_code1_high>` = 1, `<cr_gpio_dma_park_value>` = 1, `<cr_gpio_dma_out_sel_latch>` = 0 时, 对应的波

形图如下所示：

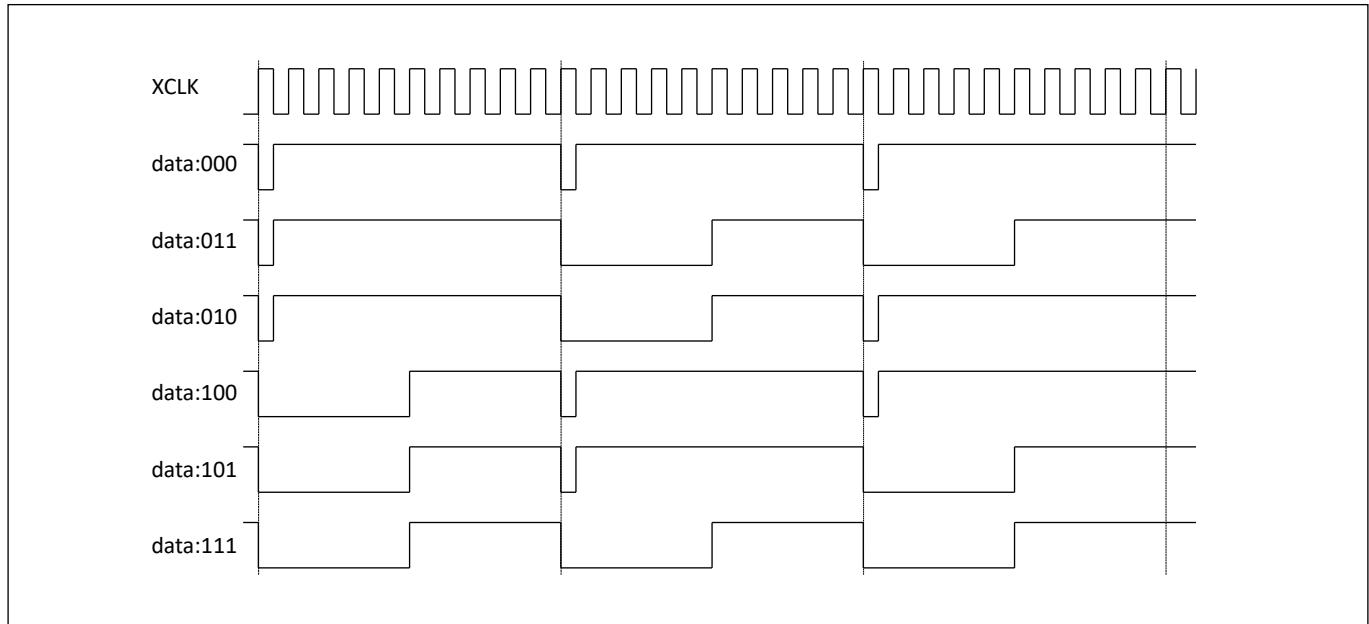


图 4.4: 默认电平为高电平, 逻辑 0 电平翻转, 逻辑 1 电平翻转输出波形

#### 4.4.4 可编程 Set/Clear 输出模式

- 将 `<reg_gpio_xx_oe>` 设置为 1, 使能 GPIO 输出模式
- 将 `<reg_gpio_xx_func_sel>` 设置为 11, 进入 SW-GPIO 模式
- 并且将 `<reg_gpio_xx_mode>` 设置为 3, 使能 I/O 的缓冲 Set/Clear 输出功能
- 可通过 `<reg_gpio_xx_pu>` 和 `<reg_gpio_xx_pd>` 设置是否使能内部上拉和下拉功能

此时当 `GPIO_CFG142` 寄存器的位 `<cr_gpio_tx_en>` 为 1 时, 由 `GPIO_CFG144` 寄存器写入 FIFO 的数据将被按顺序逐个设置对应引脚的电平, 缓冲区大小为 128 \* 16bits。

在此模式下输出到引脚的电平状态可以自由设置。以 `XCLK` 为时钟源, `GPIO_CFG142` 寄存器中位 `<cr_code_total_time>` 设置的数值为一个周期:

每 8 个引脚为一组, `GPIO_CFG144` 寄存器低 8 位和高 8 位分别设置这 8 个引脚输出高/低电平。若低 8 位写 1, 则对应的引脚输出高电平; 若高 8 位写 1, 则对应的引脚输出低电平, 此时该寄存器中写 0 的位不生效, 当高 8 位和低 8 位中对应位同时将同一个引脚写 1 时, 该引脚电平为高电平。

此外, `GPIO_CFG143` 寄存器中的位 `<cr_gpio_dma_out_sel_latch>` 应该设置为 1。`<cr_gpio_dma_park_value>` 用于设置 I/O 默认电平, 为 1 时默认电平是高电平, 为 0 时默认电平是低电平。

## 4.5 I/O FIFO

I/O FIFO 的深度为 128 半字，GPIO\_CFG143 寄存器中的位 `<gpio_tx_fifo_cnt>` 表示 FIFO 当前可用空间的大小，默认值是 128，每次向 GPIO\_CFG144 寄存器写入一个数值后，`<gpio_tx_fifo_cnt>` 的值都会递减 1，如果减至 0 后继续向 GPIO\_CFG144 寄存器写入数值，且 `<cr_gpio_tx_fer_en>` 为 1 即错误中断被使能，则会产生该中断。

当 GPIO\_CFG142 寄存器的位 `<cr_gpio_tx_en>` 为 1 时，I/O FIFO 的数据被逐个发送到 I/O 引脚，此时 `<gpio_tx_fifo_cnt>` 的值会递增，当递增到大于 `<cr_gpio_tx_fifo_th>` 时，且 `<cr_gpio_tx_fifo_en>` 为 1 即 FIFO 中断被使能，则会产生该中断。

如果 CR\_GPIO\_CFG143 寄存器的位 `<cr_gpio_dma_tx_en>` 为 1，则使能 DMA 发送数据，此时如果 `<cr_gpio_tx_fifo_th>` 小于 `<gpio_tx_fifo_cnt>`，则 DMA 会将数据从设定好的 RAM 中搬运至缓冲区，此时中断标志 `<r_gpio_tx_fifo_int>` 自动被清除。

## 4.6 I/O 中断

I/O 支持多种外部中断，将 GPIO\_CFGxx 寄存器的 `<reg_gpio_xx_int_mask>` 设置为 0，即可使能对应引脚的外部中断功能，`<reg_gpio_xx_int_mode_set>` 用于设置对应引脚的外部中断类型。支持的中断类型如下：

- 同步下降沿中断
  - 以 f32k\_clk 时钟为基准，在每个时钟上升沿采样一次输入引脚电平，若出现一个高电平后紧跟两个低电平，则此时产生同步下降沿中断
- 同步上升沿中断
  - 以 f32k\_clk 时钟为基准，在每个时钟上升沿采样一次输入引脚电平，若出现一个低电平后紧跟两个高电平，则此时产生同步上升沿中断
- 同步低电平中断
  - 以 f32k\_clk 时钟为基准，检测到低电平后，在第三个时钟上升沿处产生同步低电平中断
- 同步高电平中断
  - 以 f32k\_clk 时钟为基准，检测到高电平后，在第三个时钟上升沿处产生同步高电平中断
- 同步双边沿中断
  - 以 f32k\_clk 时钟为基准，若检测到高电平转换至低电平（低电平转换至高电平），会产生下降沿（上升沿）事件，事件产生后在第三个时钟上升沿处产生同步双边沿中断
- 异步下降沿中断
  - 当检测到高电平转换至低电平时，立即触发异步下降沿中断
- 异步上升沿中断
  - 当检测到低电平转换至高电平时，立即触发异步上升沿中断

- 异步低电平中断

- 以 f32k\_clk 时钟为基准，在每个时钟上升沿采样一次输入引脚电平，若出现连续 3 次都为低电平，则触发异步低电平中断

- 异步高电平中断

- 以 f32k\_clk 时钟为基准，在每个时钟上升沿采样一次输入引脚电平，若出现连续 3 次都为高电平，则触发异步高电平中断

在中断函数中可以通过 GPIO\_CFGxx 寄存器 <gpio\_xx\_int\_stat> 获取到产生中断的 GPIO 状态，同时可以通过 <reg\_gpio\_xx\_int\_clr> 清除对应的中断标志。

## 5.1 简介

芯片内置一个 12 bits 的逐次逼近式模拟数字转换器 (ADC)，支持 12 路外部模拟输入和若干内部模拟信号选择。ADC 可以工作在单次转换和多通道扫描两种模式下，转换结果为 12/14/16 bits 左对齐模式。ADC 拥有深度为 32 的 FIFO，支持多种中断，支持 DMA 操作。ADC 除了用于普通模拟信号测量外，还可以用于测量供电电压，此外 ADC 还可以通过测量内/外部二极管电压用于温度检测。

## 5.2 主要特点

- 高性能
  - 可以选择 12-bit, 14-bit, 16-bit 转换结果输出
  - ADC 转换时间最快 0.5us (12-bit 转换结果)
  - 支持 1.8V, 3.3V 可选参考电压
  - 支持 DMA 将转换结果搬运到内存
  - 支持单通道转换和多通道扫描两种模式
  - 支持单端与差分两种输入模式
  - 支持抖动补偿
  - 支持用户自行设定转换结果偏移值
  - 扫描模式时钟最大支持 1M，非扫描模式支持 2M
- 模拟通道数
  - 12 个外部模拟通道
  - 2 个 DAC 内部通道

- 1 个 VBAT/2 通道
- 1 个 TSEN 通道

### 5.3 功能描述

ADC 模块基本框图如图所示。

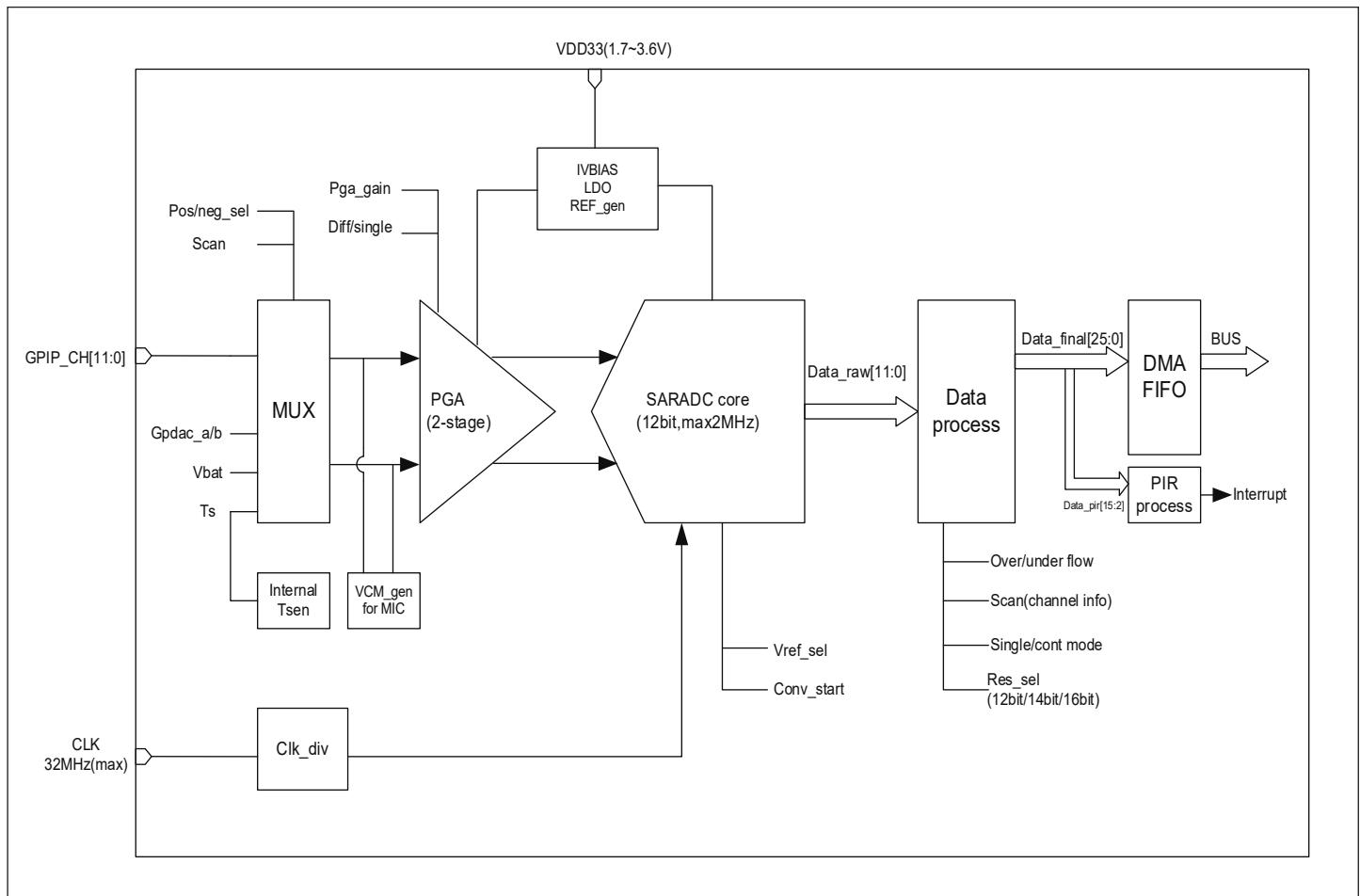


图 5.1: ADC 基本框图

ADC 模块包含五大部分，分别为前端输入通道选择器，程控放大器，ADC 采样模块，数据处理模块以及 FIFO。

输入通道选择器用于选择需要采样的通道，既包含外部模拟信号，也包含内部模拟信号。

程控放大器用于对输入信号做进一步处理，根据输入信号的特点（直流/交流）进行设定，以便得到更准确的转换值。

ADC 采样模块是最主要的功能模块，实现通过逐次比较的方式，得到模拟信号到数字信号的转换，转换结果为 12 bit。

数据处理模块负责将转换的结果进一步处理，包括添加通道信息等步骤。

最后得到的数据会推送到最后端的 FIFO 中。

### 5.3.1 ADC 引脚和内部信号

表 5.1: ADC 内部信号

内部信号	信号类型	信号描述
VBAT/2	Input	从电源引脚分压过来的电压信号
TSEN	Input	内部温度传感器输出电压
VREF	Input	内部模拟模块参考电压
DACOUTA	Input	DAC 模块输出
DACOUTB	Input	DAC 模块输出

表 5.2: ADC 外部引脚

外部引脚	信号类型	信号描述
VDDA	Input	模拟模块供电电压正极
VSSA	Input	模拟模块供电地
ADC_CHX	Input	模拟输入引脚, 总共 12 路

### 5.3.2 ADC 通道

ADC 采样的可以选择的通道包括外部模拟引脚的输入信号和芯片内部可选信号, 具体包括:

- ADC CH0
- ADC CH1
- ADC CH2
- ADC CH3
- ADC CH4
- ADC CH5
- ADC CH6
- ADC CH7
- ADC CH8
- ADC CH9
- ADC CH10

- ADC CH11
- DAC OUTA
- DAC OUTB
- VBAT/2
- TSEN
- VREF
- GND

需要注意的是，如果选择 VBAT/2 或 TSEN 作为输入待采信号，需要把 `gpadc_vbat_en` 或 `gpadc_ts_en` 置位。ADC 模块可以支持单端输入或者差分输入，如果是单端输入模式，负极输入通道需要选择 GND。

### 5.3.3 ADC 时钟

ADC 模块的工作时钟来源如下图所示。

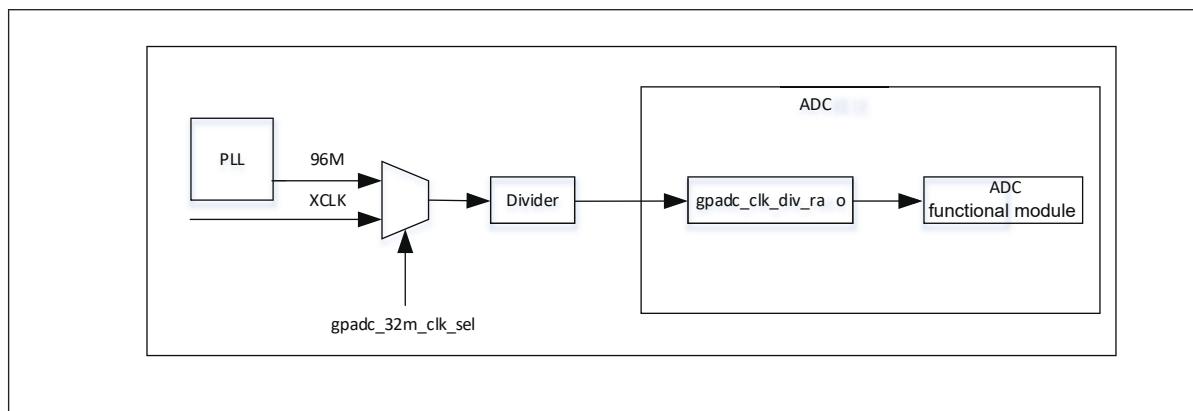


图 5.2: ADC 时钟

ADC 的时钟源可以选择来自 PLL 的 96M, XTAL 或者内部 RC32M, 时钟源的选择在 GLB 模块中设定，同时 GLB 模块也提供了时钟分频，默认情况下，ADC 的时钟源是 96M，时钟分频是 2，到达 ADC 模块的时钟是 32M。在 ADC 模块内部，提供了一个时钟分频，默认 16 分频，故 ADC 模块内部的时钟默认是 2M。用户可以根据实际采样需求，自行调整 ADC 的时钟源和各个分频系数。`gpadc_32m_clk_div` 分频寄存器宽度为 6bits，最大分频为 64，分频公式为  $f_{out} = f_{source} / (gpadc_32m_clk_div + 1)$ 。`gpadc_clk_div_ratio` 分频寄存器位于 ADC 模块内部，宽度为 3bits，其分频值定义如下：

- 3'b000: div=1
- 3'b001: div=4
- 3'b010: div=8
- 3'b011: div=12

- 3'b100: div=16
- 3'b101: div=20
- 3'b110: div=24
- 3'b111: div=32

### 5.3.4 ADC 转换模式

ADC 支持单通道转换和扫描转换两种模式。

- 单通道转换模式
  - 通过设置 `gpadc_reg_cmd` 寄存器的 `<gpadc_pos_sel>` 选择正极输入通道, `<gpadc_neg_sel>` 选择负极输入通道
  - 将 `gpadc_reg_config1` 寄存器的 `<gpadc_cont_conv_en>` 控制位设置为 0, 表示单通道转换,

然后设置 `gpadc_conv_start` 控制位启动转换即可。

在扫描转换模式下, `gpadc_cont_conv_en` 控制位需要设置为 1, ADC 根据 `gpadc_scan_length` 控制位设定的转换通道个数, 依次按照 `gpadc_reg_scn_posX(X=1, 2)` 和 `gpadc_reg_scn_negX(X=1, 2)` 寄存器组所设定的通道顺序, 逐个进行转换, 转换的结果会自动推入 ADC 的 FIFO。`gpadc_reg_scn_posX(X=1, 2)` 和 `gpadc_reg_scn_negX(X=1, 2)` 寄存器组所设定的通道可以相同, 这也就意味着用户可以实现对一个通道进行多次采样转换。

ADC 的转换结果一般都是放入 FIFO 中。用户需要根据实际转换通道数, 设定 FIFO 接收数据阈值中断, 通过 FIFO 的阈值中断, 作为 ADC 转换完成中断。

### 5.3.5 ADC 结果

`gpadc_raw_data` 寄存器存放了 ADC 的原始结果, 在单端模式下, 数据有效位是 12bits, 无符号位, 在差分模式下, 最高位为符号位, 剩下 11bits 代表转换的结果。

`gpadc_data_out` 寄存器存放了 ADC 的结果, 这个结果里包含了 ADC 结果, 符号位和通道信息, 数据格式如下:

表 5.3: ADC 转换结果含义

BitS	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义	正极通道号				负极通道号				转换结果																	

转换结果的 bit21-bit25 是正极通道号, bit16-bit20 是负极通道号, bit0-bit15 是转换的数值。

`gpadc_res_sel` 控制位可以设定转换结果的位数为 12 位, 14 位, 和 16 位, 其中 14 位和 16 位是多次采样提高精度得到的结果, 其可以设置的值如下:

- 3'b000 12bit 2MS/s, OSR=1

- 3'b001 14bit 125kS/s, OSR=16
- 3'b010 14bit 31.25kS/s, OSR=64
- 3'b011 16bit 15.625KS/s, OSR=128
- 3'b100 16bit 7.8125KS/s, OSR=256

ADC 转换结果为左对齐模式，当选择 12 位时，转换结果的 bit15-bit4 有效，当选择 14 位时，转换结果的 bit15-bit2 有效，当选择 16 位时，转换结果的 bit15-bit0 有效。同样，在差分模式下，最高位是符号位，也就是，当选择 14 位时，bit15 是符号位，bit14-bit2 是转换结果，bit14 是 MSB，在单端模式下，没有符号位，也就是，当选择 12 位时，bit15-bit4 是转换结果，bit15 是 MSB。

在实际使用中，ADC 的结果一般都是放入 FIFO，这在多通道扫描模式下尤为重要，所以用户一般都是从 ADC FIFO 获取转换结果，ADC FIFO 的数据格式 `gpadc_data_out` 寄存器中数据格式相同。

### 5.3.6 ADC 中断

ADC 模块在正极采样超量程和负极采样超量程时可以产生中断，可以通过 `gpadc_pos_satur_mask`,`gpadc_neg_satur_mask` 屏蔽各自中断，当中断产生时，可以通过 `gpadc_pos_satur`, 和 `gpadc_neg_satur` 寄存器查询中断状态，同时可以通过 `gpadc_pos_satur_clr` 和 `gpadc_neg_satur_clr` 清除中断。该功能可以用来判断输入电压是否异常。

### 5.3.7 ADC FIFO

ADC 模块拥有深度为 32 的 FIFO，数据宽度为 26bits，当 ADC 完成转换后，会自动将结果推入到 FIFO。ADC 的 FIFO 有如下状态和中断管理功能：

- FIFO 满状态
- FIFO 非空状态
- FIFO Overrun 中断
- FIFO Underrun 中断

当中断产生时，可以通过对应的 `clear` 位将中断标志清除掉。

利用 ADC 的 FIFO 用户可以实现三种模式获取数据：查询模式，中断模式，DMA 模式

查询模式

CPU 轮询 `gpadc_rdy` 位，当该控制位置位时，说明 FIFO 中存在有效数据，CPU 可以根据 `gpadc_fifo_data_count` 获知 FIFO 数据个数并从 FIFO 读出这些数据。

中断模式

CPU 设置 `gpadc_rdy_mask` 为 0，ADC 就会在 FIFO 有数据推入的时候产生中断，用户可在中断函数中，根据 `gpadc_fifo_data_count` 获知 FIFO 数据个数并从 FIFO 读出这些数据，然后设置 `gpadc_rdy_clr` 清除中断。

DMA 模式

用户设定 `gpadc_dma_en` 控制位，可以配合 DMA 完成转换数据到内存的搬运，在使用 DMA 模式时，通过 `gpadc_fifo_thl` 设置 ADC FIFO 发送 DMA 请求的数据个数阈值，DMA 在收到请求时，会自动根据用户设定的参数，从 FIFO 搬运指定个数的结果到对应的内存。

### 5.3.8 ADC 设置流程

设置 **ADC** 时钟

根据 ADC 转换速度需求，确定 ADC 的工作时钟，设定 GLB 模块的 ADC 时钟源和分频，结合 `gpadc_clk_div_ratio`，确定最终 ADC 模块的工作时钟频率。

根据使用的通道设置 **GPIO**

根据使用的模拟引脚，确定使用的通道号，初始化对应的 GPIO 为模拟功能，需要注意的是，在设定 GPIO 为模拟输入的时候，不要设置 GPIO 的上拉或者下拉，需要设置为浮空输入。

设定要转换的通道

根据使用的模拟通道和转换模式，设定对应的通道寄存器，对于单通道转换，在 `gpadc_pos_sel` 和 `gpadc_neg_sel` 寄存器中设置转换的通道信息。对于多通道扫描模式，根据要扫描通道数目和扫描顺序，设定 `gpadc_scan_length`,`gpadc_reg_scn_posX` 和 `gpadc_reg_scn_negX`。

设定数据读取方式

根据 ADC FIFO 介绍的读取数据方式，选择使用的模式，设置对应的寄存器。如果使用 DMA，同样需要配置 DMA 的一个通道，配合 ADC FIFO 完成数据的搬运。

启动转换

最后设置 `gpadc_res_sel` 选择数据转换结果的精度，最后设置 `gpadc_global_en=1`, `gpadc_conv_start=1` 就可以启动 ADC 开始转换。当转换完成，需要再次转换时，需要将 `gpadc_conv_start` 设置为 0，再设置为 1，以便再次触发转换。

### 5.3.9 VBAT 测量

这里的 VBAT/2 测量的是芯片 VDD33 的电压，而不是外部的比如锂电池的电压，如果需要测量锂电池等供电源头的电压，可以将电压分压，然后输入 ADC 的 GPIO 模拟通道，测量 VDD33 的电压可以减少 GPIO 的使用。

ADC 模块测量的 VBAT/2 电压是经过分压的，实际输入到 ADC 模块的电压是 VDD33 的一半，即  $VBAT/2=VDD33/2$ 。由于电压经过分压，为了得到较高的精确度，建议 ADC 的参考电压选择 1.8V，采用单端模式，正极输入电压选择 VBAT/2，负极输入电压选择 GND，同时将 `Gpadc_vbat_en` 设置为 1，启动转换后，将对应的转换结果乘以 2 就可以得到 VDD33 电压。

### 5.3.10 TSEN 测量

ADC 可以测量内部二极管或者外部二极管电压值，而二极管的压差和温度有关，所以通过测量二极管的电压，可以计算得到环境温度，我们称之为 Temperature Sensor，简称 TSEN。

TSEN 的测试原理是通过一个二极管上面测量两次不同大小的电流产生的电压差  $\Delta V$  随着温度的变化拟合的曲线，无论外部或者内部二极管的测量，最终输出的值和温度有关，都可以表示成  $\Delta(ADC\_out)=7.753T+X$ ，当我们知道了电压值，也就知道了温度  $T$ 。这里的  $X$  是一个偏移值，可以作为标准值，在实际使用前，我们需要确定  $X$ 。芯片厂商会在芯片出厂前，在标准温度下，例如室温 25 度，测量  $\Delta(ADC\_out)$ ，从而得到  $X$ 。在用户使用的时候，只要根据公式  $T=[\Delta(ADC\_out)-X]/7.753$ ，就可以得到温度  $T$ 。

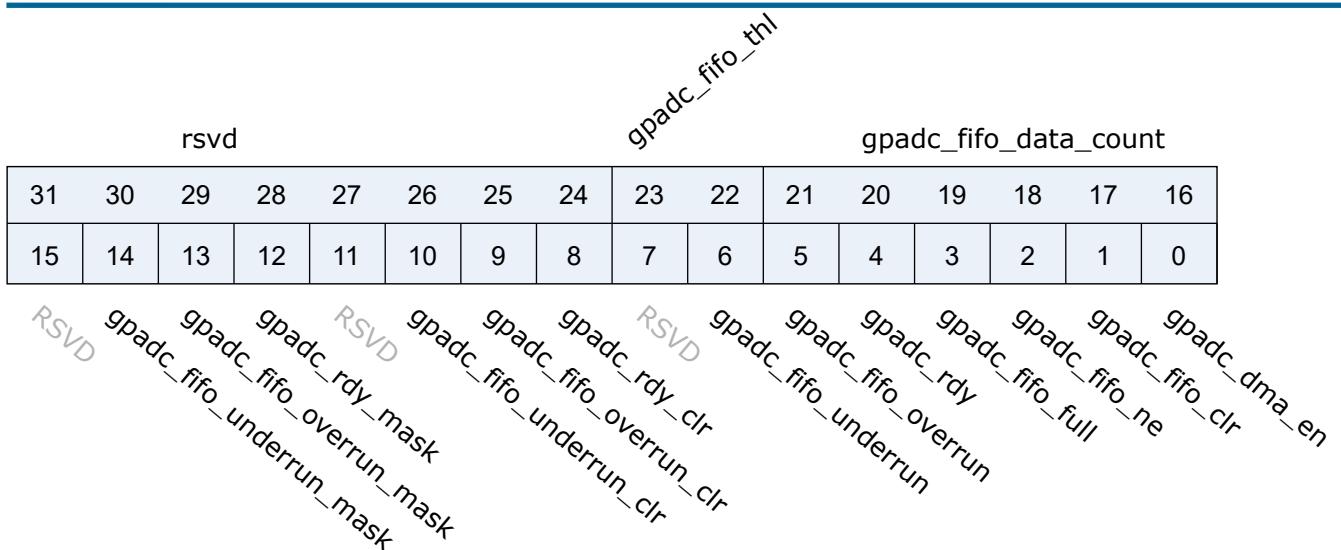
在使用 TSEN 时，建议把 ADC 设置成 16bits 模式，通过多次采样以减少误差，参考电压选择 1.8V 以提高精度，设置 `gpadc_ts_en` 为 1 以便启动 TSEN 功能，如果选择内部二极管，`gpadc_tsext_sel=0`，如果选择外部二极管，`gpadc_tsext_sel=1`，根据实际情况选择正向输入通道，如果是内部二极管，选择 TSEN 通道，如果是外部，选择对应的模拟 GPIO 通道，负极输入端选择 GND。在上述设定完毕后，设置 `gpadc_tsvbe_low=0`，启动测量，得到测量结果  $V_0$ ，再设置 `gpadc_tsvbe_low=1`，启动测量，得到测量结果  $V_1$ ， $\Delta(ADC\_out)=V_1-V_0$ ，根据公式  $T=[\Delta(ADC\_out)-X]/7.753$ ，得到温度  $T$ 。

## 5.4 寄存器描述

名称	描述
<code>gpadc_config</code>	
<code>gpadc_dma_rdata</code>	
<code>gpadc_pir_train</code>	

### 5.4.1 gpadc\_config

地址：0x20002000



位	名称	权限	复位值	描述
31:24	rsvd	rsvd	8'd0	
23:22	gpadc_fifo_thl	r/w	2'd0	fifo threshold 2'b00: 1 data 2'b01: 4 data 2'b10: 8 data 2'b11: 16 data
21:16	gpadc_fifo_data_count	r	6'd0	fifo data number
15	RSVD			
14	gpadc_fifo_underrun_mask	r/w	1'b0	write 1 mask
13	gpadc_fifo_overrun_mask	r/w	1'b0	write 1 mask
12	gpadc_rdy_mask	r/w	1'b0	write 1 mask
11	RSVD			
10	gpadc_fifo_underrun_clr	r/w	1'b0	Write 1 to clear flag
9	gpadc_fifo_overrun_clr	r/w	1'b0	Write 1 to clear flag
8	gpadc_rdy_clr	r/w	1'b0	Write 1 to clear flag
7	RSVD			
6	gpadc_fifo_underrun	r	1'b0	FIFO underrun interrupt flag
5	gpadc_fifo_overrun	r	1'b0	FIFO overrun interrupt flag
4	gpadc_rdy	r	1'b0	Conversion data ready interrupt flag
3	gpadc_fifo_full	r	1'b0	FIFO full flag
2	gpadc_fifo_ne	r	1'b0	FIFO not empty flag

位	名称	权限	复位值	描述
1	gpadc_fifo_clr	w1c	1'b0	FIFO clear signal
0	gpadc_dma_en	r/w	1'b0	GPADC DMA enable

#### 5.4.2 gpadc\_dma\_rdata

地址: 0x20002004

gpadc_dma_rdata															
rsvd								gpadc_dma_rdata							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpadc\_dma\_rdata

位	名称	权限	复位值	描述
31:26	rsvd	rsvd	6'd0	
25:0	gpadc_dma_rdata	r	26'd0	GPADC final conversion result stored in the FIFO

#### 5.4.3 gpadc\_pir\_train

地址: 0x20002020

pir_train															
RSVD								pir_stop							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pir\_cnt\_v

pir\_extend

位	名称	权限	复位值	描述
31:18	RSVD			
17	pir_stop	r	0	PIR Training End
16	pir_train	r/w	0	PIR Training Mode
15:13	RSVD			
12:8	pir_cnt_v	r	0	GPADC Record Extension Counter Value
7:5	RSVD			

位	名称	权限	复位值	描述
4:0	pir_extend	r/w	5'd15	GPADC Record Extension after PIR interrupt

## 6.1 简介

芯片内置一个 10bits 的数字模拟转换器 (DAC) ,FIFO 深度为 1，支持 2 路 DAC 调制输出。可用于音频播放，变送器电压调制。

## 6.2 主要特点

- DAC 调制精度为 10-bits
- DAC 的输入时钟可选为 32k、16k、8k 或 512k
- 支持 DMA 将内存搬运至 DAC 调制寄存器
- 支持双声道播放 DMA 搬运模式
- DAC 的输出引脚固定为 ChannelA 为 GPIO11,ChannelB 为 GPIO4

## 6.3 功能描述

DAC 模块基本框图如图所示。

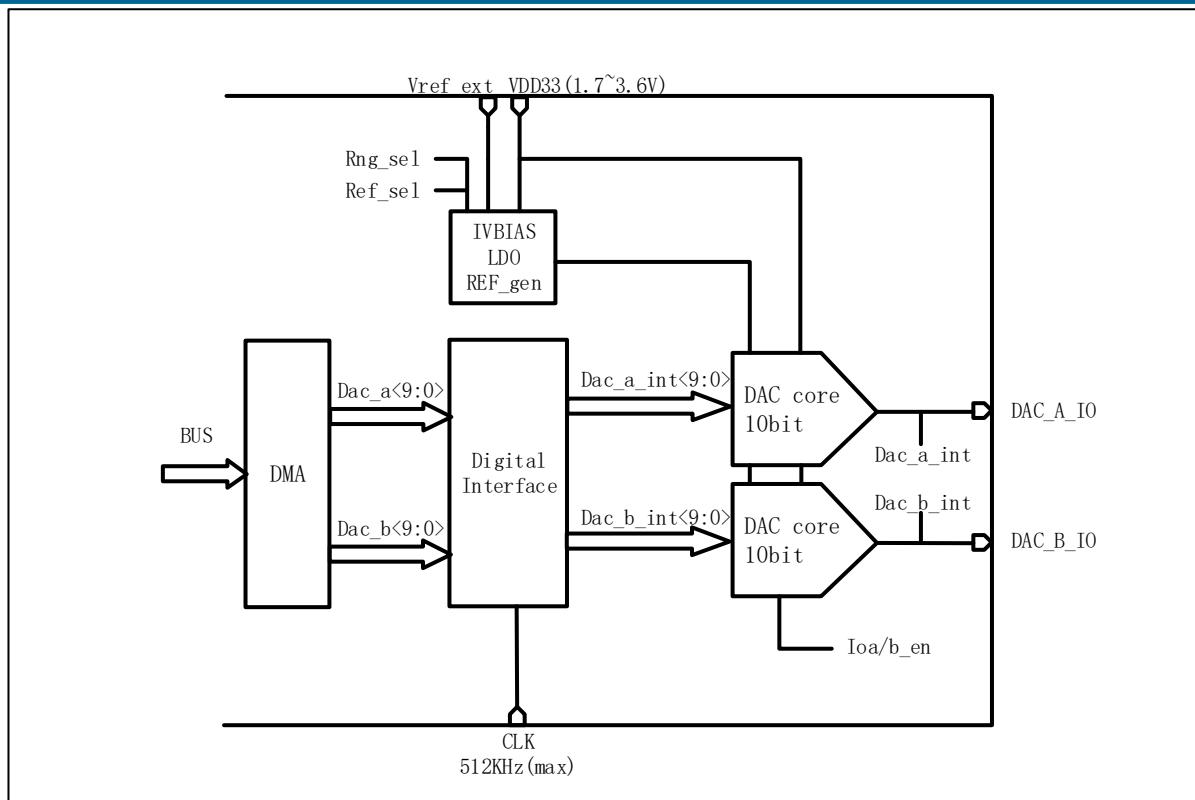


图 6.1: DAC 基本框图

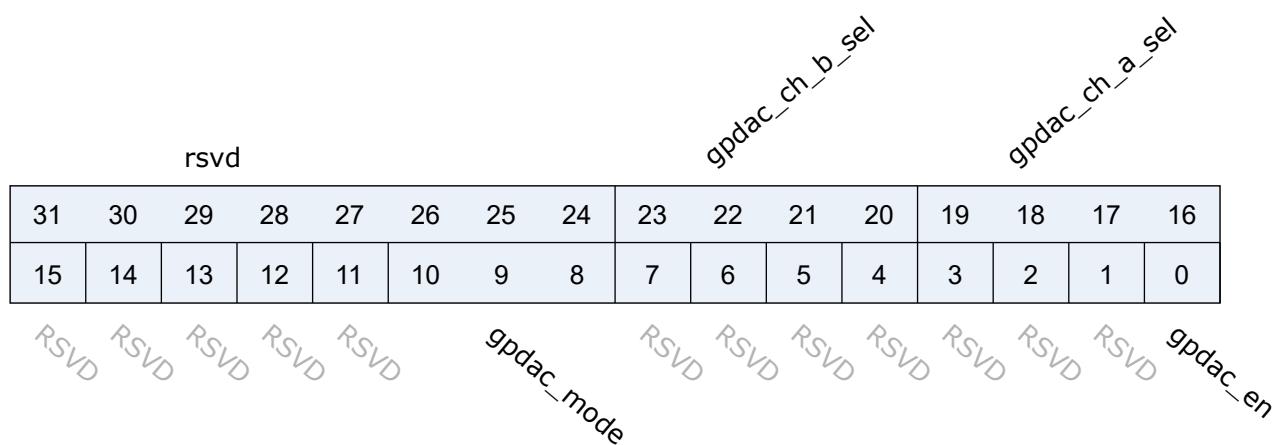
- DAC 模块支持最多两路调制输出
- DAC 模块支持双声道 DMA 数据搬运模式
- DAC 模块支持长度为 32-bit 的 DMA 数据接口，其中高 16 位将会调制在 ChannelA 的引脚上，低 16 位调制在 ChannelB 引脚。

## 6.4 寄存器描述

名称	描述
gpdac_config	
gpdac_dma_config	
gpdac_dma_wdata	
gpdac_tx_fifo_status	

### 6.4.1 gpdac\_config

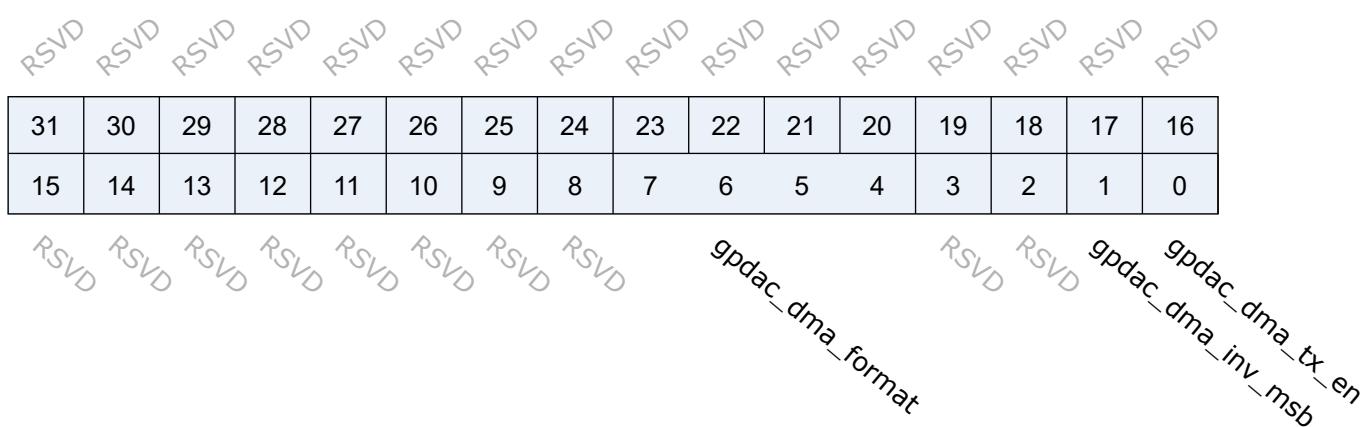
地址: 0x20002040



位	名称	权限	复位值	描述
31:24	rsvd	rsvd	8'h0d	
23:20	gpdac_ch_b_sel	r/w	0	Channel B Source Select 0: Reg 1: DMA 2: DMA + Filter 3: Sin Gen 4: A (The same as channel A) 5: A (Inverse of channel A)
19:16	gpdac_ch_a_sel	r/w	0	Channel A Source Select 0: Reg 1: DMA 2: DMA + Filter 3: Sin Gen
15:11	RSVD			
10:8	gpdac_mode	r/w	0	0:32k, 1:16k, 3:8k, 4:512k(for DMA only)
7:1	RSVD			
0	gpdac_en	r/w	0	GPDAC enable

## 6.4.2 gpdac\_dma\_config

地址: 0x20002044



位	名称	权限	复位值	描述
31:8	RSVD			
7:4	gpdac_dma_format	r/w	0	DMA TX format (Data 10-bit) 0: ([9:0]) A0, A1, A2... 1: ([25:16][9:0]) B0,A0, B1,A1, B2,A2... 2: ([25:16][9:0]) A1,A0, A3,A2, A5,A4... 4: ([15:6]) A0, A1, A2... 5: ([31:22][15:6]) B0,A0, B1,A1, B2,A2... 6: ([31:22][15:6]) A1,A0, A3,A2, A5,A4... 8: ([31:24][23:16][15:8][7:0]) A3,A2,A1,A0, A7,A6,A5,A4... 9: ([31:24][23:16][15:8][7:0]) B1,B0,A1,A0, B3,B2,A3,A2... 10: ([31:24][23:16][15:8][7:0]) B1,A1,B0,A0, B3,A3,B2,A2... 11: ([15:8][7:0]) B0,A0, B1,A1, B2,A2...
3:2	RSVD			
1	gpdac_dma_inv_msb	r/w	0	GPDAC DMA Data Inverse MSB
0	gpdac_dma_tx_en	r/w	0	GPDAC DMA TX enable

### 6.4.3 gpdac\_dma\_wdata

地址: 0x20002048

gpdac\_dma\_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpdac\_dma\_wdata

位	名称	权限	复位值	描述
31:0	gpdac_dma_wdata	w	x	GPDAC DMA TX data

### 6.4.4 gpdac\_tx\_fifo\_status

地址: 0x2000204c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

*RSVD RSVD RSVD RSVD RSVD TxFifoWrPtr TxFifoRdPtr tx\_cs tx\_fifo\_full tx\_fifo\_empty*

位	名称	权限	复位值	描述
31:10	RSVD			
9:8	TxFifoWrPtr	r	0	
7:4	TxFifoRdPtr	r	4'd8	
3:2	tx_cs	r	0	
1	tx_fifo_full	r	0	
0	tx_fifo_empty	r	0	

## 7.1 简介

DMA(Direct Memory Access) 是一种内存存取技术，可以独立地直接读写系统内存，而不需处理器介入处理。在同等程度的处理器负担下，DMA 是一种快速的数据传送方式。3 个独立的 DMA 控制器，其中 DMA0 和 DMA2 有 8 组独立专用通道，DMA1 有 4 组独立专用通道，管理外围设备和内存之间的数据传输以提高总线效率。主要有三种类型传输包括内存至内存、内存至外设、外设至内存。并支持 LLI 链接列表功能。使用上由软件配置传输数据大小、数据源地址和目标地址。

## 7.2 主要特征

- 3 个 DMA 控制器，分别为 DMA0、DMA1 和 DMA2
- DMA0 和 DMA2 有 8 组独立专用通道，DMA1 有 4 组独立专用通道
- 独立控制来源与目标存取宽度(单字节、双字节、四字节)
- 每个通道独立作为读写缓存
- 每个通道可被独立的外设硬件触发或是软件触发
- DMA0 和 DMA1 支持外设包括 UART、I2C、SPI、ADC、IR、GPIO、Audio、I2S、PDM
- DMA2 支持外设包括 UART、I2C、SPI、DBI、DSI
- 八种流程控制
  - DMA 流程控制，来源内存、目标内存
  - DMA 流程控制，来源内存、目标外设
  - DMA 流程控制，来源外设、目标内存
  - DMA 流程控制，来源外设、目标外设
  - 目标外设流程控制，来源外设、目标外设
  - 目标外设流程控制，来源内存、目标外设
  - 来源外设流程控制，来源外设、目标内存
  - 来源外设流程控制，来源外设、目标外设
- 支持 LLI 链表功能，提高 DMA 效率

## 7.3 功能描述

### 7.3.1 工作原理

当一个设备试图通过总线直接向另一个设备传输数据时，它会先向 CPU 发送 DMA 请求信号。外设通过 DMA 向 CPU 提出接管总线控制权的总线请求，CPU 收到该信号后，在当前的总线周期结束后，会按 DMA 信号的优先级和提出 DMA 请求的先后顺序响应 DMA 信号。CPU 对某个设备接口响应 DMA 请求时，会让出总线控制权。于是在 DMA 控制器的管理下，外设和存储器直接进行数据交换，而不需 CPU 干预。数据传送完毕后，设备会向 CPU 发送 DMA 结束信号，交还总线控制权。

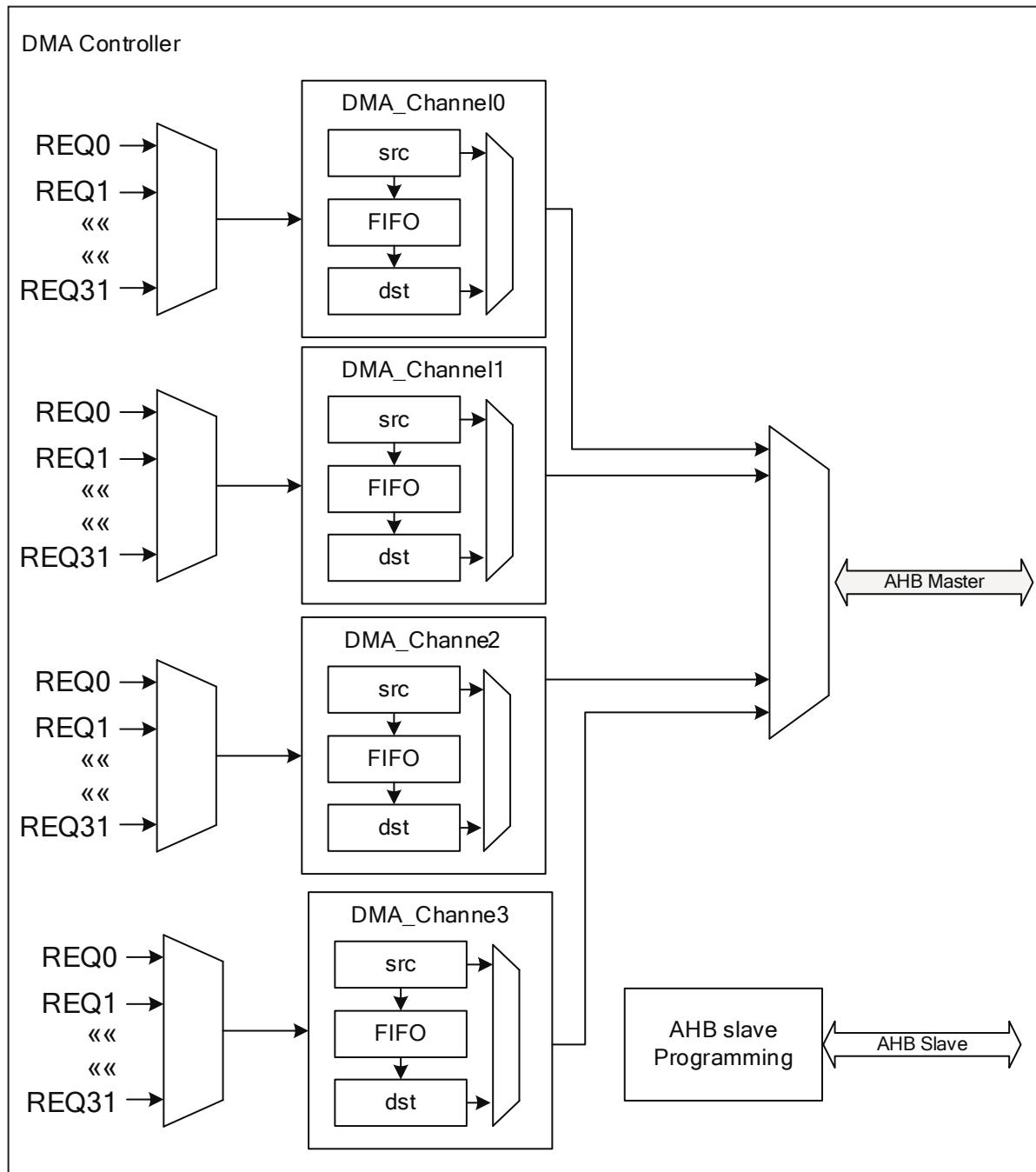


图 7.1: DMA 框图

DMA 包含一组 AHB Master 接口和一组 AHB Slave 接口。AHB Master 接口根据当前配置需求通过系统总线主动存取内存或是外设，做为数据搬移的端口。AHB Slave 接口作为配置 DMA 的接口，只支持 32-bit 存取。

### 7.3.2 DMA 通道配置

DMA0 和 DMA2 共支持 8 路通道，DMA1 支持 4 路通道，各通道之间互不干涉，可以同时运行，下面是 DMA 通道 x 的配置过程：

1. 在 DMA\_C0SrcAddr 寄存器中设置 32-bit 来源地址
2. 在 DMA\_C0DstAddr 寄存器中设置 32-bit 目标地址
3. 地址自动累加，可通过配置 DMA\_C0Control 寄存器中 SI(来源)、DI(目标) 设定是否开启地址自动累加模式，设置为 1 时，开启地址自动累加模式
4. 设置传输数据宽度，可通过配置 DMA\_C0Control 寄存器中 SWidth(来源)、DWidth(目标) 位，宽度选项有单字节、双字节、四字节、八字节 (仅 DMA2 支持)
5. Burst 型态，可通过配置 DMA\_C0Control 寄存器中 SBSIZE(来源)、DBSIZEx(目标) 位来设置，配置选项有 INCR1、INCR4、INCR8、INCR16
6. 需要特别注意的是所配置的组合，DMA0 和 DMA1 单笔 burst 不能超过 16 字节，DMA2 单笔 burst 不能超过 128 字节
7. 设置数据传输长度的范围为：0-4095

### 7.3.3 外设支持

可通过配置 SrcPeripheral(来源) 和 DstPeripheral(目标) 来决定当前 DMA 配合的外设，外设类型与配置值的对应关系如下表所示：

Value	DMA0	DMA1	DMA2
0	UART0_RX	UART0_RX	UART3_RX
1	UART0_TX	UART0_TX	UART3_TX
2	UART1_RX	UART1_RX	SPI1_RX
3	UART1_TX	UART1_TX	SPI1_TX
4	UART2_RX	UART2_RX	-
5	UART2_TX	UART2_TX	-
6	I2C0_RX	I2C0_RX	I2C2_RX
7	I2C0_TX	I2C0_TX	I2C2_TX
8	IR_TX	IR_TX	I2C3_RX
9	GPIO_TX	GPIO_TX	I2C3_TX
10	SPI0_RX	SPI0_RX	DSI_RX
11	SPI0_TX	SPI0_TX	DSI_TX
12	AUDIO_RX	AUDIO_RX	-
13	AUDIO_TX	AUDIO_TX	-
14	I2C1_RX	I2C1_RX	-
15	I2C1_TX	I2C1_TX	-
16	I2S_RX	I2S_RX	-
17	I2S_TX	I2S_TX	-
18	PDM_RX	PDM_RX	-
22	GPADC_RX	GPADC_RX	DBI_TX
23	GPADC_TX	GPADC_TX	-

图 7.2: 外设类型选择

以下是部分外设配置示例：

#### UART 使用 DMA 传输数据

UART 发送数据包，使用 DMA 方式能大量减轻 CPU 处理的时间，使其 CPU 资源不被大量浪费，尤其在 UART 收发大量数据包（如高频率收发指令）时具有明显优势。

以 UART0 传输为例，配置过程如下：

1. 将寄存器 DMA\_C0Config 中 SrcPeripheral 位的值设置为 1，即将 Source peripheral 设置为 UART0\_TX
2. 将寄存器 DMA\_C0Config 中 DstPeripheral 位的值设置为 0，即将 Destination peripheral 设置为 UART0\_RX

#### I2C 使用 DMA 传输数据

配置如下：

1. 将寄存器 DMA\_C0Config 中 SrcPeripheral 位的值设置为 7，即将 Source peripheral 设置为 I2C0\_TX

- 
- 将寄存器 DMA\_C0Config 中 DstPeripheral 位的值设置为 6，即将 Destination peripheral 设置为 I2C0\_RX

**SPI 使用 DMA 传输数据**

配置如下：

- 将寄存器 DMA\_C0Config 中 SrcPeripheral 位的值设置为 11，即将 Source peripheral 设置为 SPI0\_TX
- 将寄存器 DMA\_C0Config 中 DstPeripheral 位的值设置为 10，即将 Destination peripheral 设置为 SPI0\_RX

**ADC0/1 使用 DMA 传输数据**

配置如下：

- 将寄存器 DMA\_C0Config 中 SrcPeripheral 位的值设置为 22/23，即将 Source peripheral 设置为 GPADC0/GPADC1

#### 7.3.4 链表模式

DMA 支持链表工作模式。在进行一次 DMA 读或写操作时，可以向下一条链表中填写数据，当完成当前链表的数据传输后，通过读取 DMA\_C0LLI 寄存器的数值获取下一条链表的起始地址，直接传输下一条链表中的数据。保证 DMA 传输过程中连续不间断的工作，提高 CPU 和 DMA 的效率。

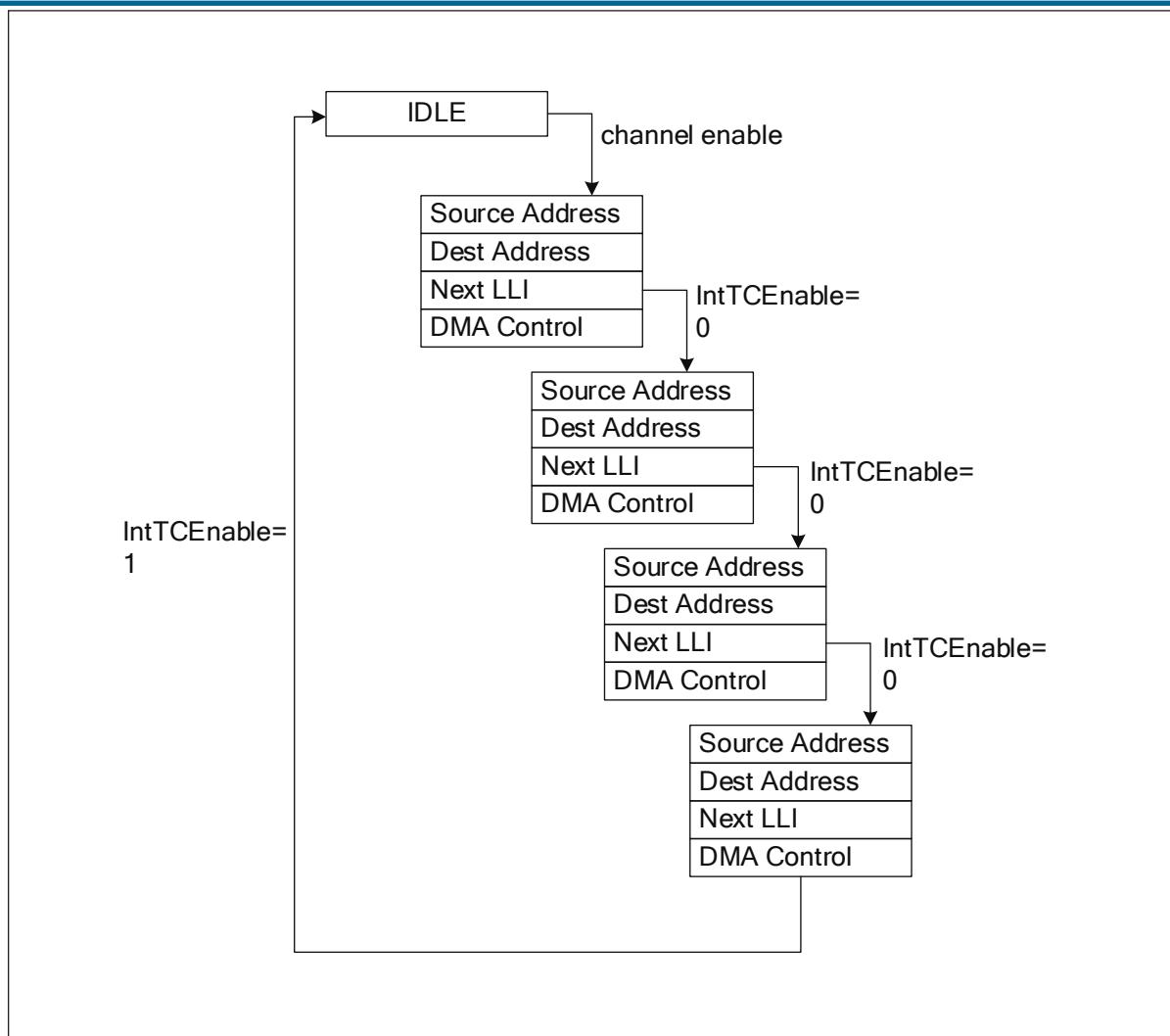


图 7.3: LLI 框架

### 7.3.5 DMA 中断

- DMA\_INT\_TCOMPLETED
  - 数据传输完成中断，当一次数据传输完毕后，会进入此中断
- DMA\_INT\_ERR
  - 数据传输出错中断，当数据传输过程中出现错误时，会进入此中断

## 7.4 传输模式

### 7.4.1 内存到内存

这个模式启动后，DMA 会根据设定好的搬移数量 (TransferSize)，将数据从来源地址搬到目标地址，传输完毕后 DMA 控制器会自动回到空闲状态，等待下一次的搬运。

具体配置流程如下：

1. 将寄存器 DMA\_C0SrcAddr 的值设置为来源的内存地址
2. 将寄存器 DMA\_C0DstAddr 的值设置为目的地的内存地址
3. 选择传输模式，将寄存器 DMA\_C0Config 中 FlowCntrl 位的值设置为 0，即选择 memory-to-memory 模式
4. 设置 DMA\_C0Control 寄存器中对应的位的数值：DI、SI 位设置为 1，开启地址自动累加模式，DWidth、SWidth 位分别设置来源和目的地的传输宽度，DBSize、SBSize 位分别设置来源和目的地的 burst 型态
5. 选择合适的通道，使能 DMA，完成数据传输

### 7.4.2 内存到外设

在这种工作模式下，DMA 会根据设定好的搬移数量 (TransferSize)，把数据从来源端搬至内部缓存，当缓存空间不够时自动暂停，待有足够的缓存空间时继续，直到设定的搬移数量达到。另外一方面当目标外设请求触发会将目标配置 burst 到目标地址，直达到设定搬移数量完成自动回到空闲状态，等待下一次启动

具体配置流程如下：

1. 将寄存器 DMA\_C0SrcAddr 的值设置为来源的内存地址
2. 将寄存器 DMA\_C0DstAddr 的值设置为目的地的外设地址
3. 选择传输模式，将寄存器 DMA\_C0Config 中 FlowCntrl 位的值设置为 1，即选择 Memory-to-peripheral 模式
4. 设置 DMA\_C0Control 寄存器中对应的位的数值：DI、SI 位设置为 1，开启地址自动累加模式，DWidth、SWidth 位分别设置来源和目的地的传输宽度，DBSize、SBSize 位分别设置来源和目的地的 burst 型态
5. 选择合适的通道，使能 DMA，完成数据传输

### 7.4.3 外设到内存

在这种工作模式下，当来源外设请求触发时将来源配置 burst 到缓存，直到设定的搬移数量达到停止。另外一方面，当内部缓存足够一次目标 burst 数量时，DMA 会自动将缓存的内容搬到目标地址直达到设定搬移数量完成自动回到空闲状态，等待下一次启动

具体配置流程如下：

1. 将寄存器 DMA\_C0SrcAddr 的值设置为来源的外设地址
2. 将寄存器 DMA\_C0DstAddr 的值设置为目的地的内存地址
3. 选择传输模式，将寄存器 DMA\_C0Config 中 FlowCntrl 位的值设置为 2，即选择 Peripheral-to-memory 模式
4. 设置 DMA\_C0Control 寄存器中对应的位的数值：DI、SI 位设置为 1，开启地址自动累加模式，DWidth、SWidth 位分别设置来源和目的地的传输宽度，DBSize、SBSize 位分别设置来源和目的地的 burst 型态
5. 选择合适的通道，使能 DMA，完成数据传输

## 7.5 寄存器描述

名称	描述
DMA_IntStatus	
DMA_IntTCStatus	

名称	描述
DMA_IntTCClear	
DMA_IntErrorStatus	
DMA_IntErrClr	
DMA_RawIntTCStatus	
DMA_RawIntErrorStatus	
DMA_EnbldChns	
DMA_SoftBReq	
DMA_SoftSReq	
DMA_SoftLBReq	
DMA_SoftLSReq	
DMA_Config	
DMA_Sync	
DMA_C0SrcAddr	
DMA_C0DstAddr	
DMA_C0LLI	
DMA_C0Control	
DMA_C0Config	
DMA_C0RSVD	
DMA_C1SrcAddr	
DMA_C1DstAddr	
DMA_C1LLI	
DMA_C1Control	
DMA_C1Config	
DMA_C1RSVD	
DMA_C2SrcAddr	
DMA_C2DstAddr	
DMA_C2LLI	
DMA_C2Control	
DMA_C2Config	
DMA_C2RSVD	
DMA_C3SrcAddr	

名称	描述
DMA_C3DstAddr	
DMA_C3LLI	
DMA_C3Control	
DMA_C3Config	
DMA_C3RSVD	
DMA_C4SrcAddr	
DMA_C4DstAddr	
DMA_C4LLI	
DMA_C4Control	
DMA_C4Config	
DMA_C4RSVD	
DMA_C5SrcAddr	
DMA_C5DstAddr	
DMA_C5LLI	
DMA_C5Control	
DMA_C5Config	
DMA_C5RSVD	
DMA_C6SrcAddr	
DMA_C6DstAddr	
DMA_C6LLI	
DMA_C6Control	
DMA_C6Config	
DMA_C6RSVD	
DMA_C7SrcAddr	
DMA_C7DstAddr	
DMA_C7LLI	
DMA_C7Control	
DMA_C7Config	
DMA_C7RSVD	

### 7.5.1 DMA\_IntStatus

地址: 0x30001000

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

IntStatus

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntStatus	r	0	Status of the DMA interrupts after masking

### 7.5.2 DMA\_IntTCStatus

地址: 0x30001004

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

IntTCStatus

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntTCStatus	r	0	Interrupt terminal count request status

### 7.5.3 DMA\_IntTCClear

地址: 0x30001008

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

IntTCClear

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntTCClear	w	0	Terminal count request clear

#### 7.5.4 DMA\_IntErrorStatus

地址: 0x3000100c

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
															IntErrorStatus

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntErrorStatus	r	0	Interrupt error status

#### 7.5.5 DMA\_IntErrClr

地址: 0x30001010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
															IntErrClr

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntErrClr	w	0	Interrupt error clear

### 7.5.6 DMA\_RawIntTCStatus

地址: 0x30001014

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RawIntTCStatus

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	RawIntTCStatus	r	0	Status of the terminal count interrupt prior to masking

### 7.5.7 DMA\_RawIntErrorStatus

地址: 0x30001018

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RawIntErrorStatus

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	RawIntErrorStatus	r	0	Status of the error interrupt prior to masking

### 7.5.8 DMA\_EnbldChns

地址: 0x3000101c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

EnabledChannels

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	EnabledChannels	r	0	Channel enable status

### 7.5.9 DMA\_SoftBReq

地址: 0x30001020

SoftBReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftBReq

位	名称	权限	复位值	描述
31:0	SoftBReq	r/w	0	Software burst request

### 7.5.10 DMA\_SoftSReq

地址: 0x30001024

SoftSReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftSReq

位	名称	权限	复位值	描述
31:0	SoftSReq	r/w	0	Software single request

### 7.5.11 DMA\_SoftLBReq

地址: 0x30001028

SoftLBReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftLBReq

位	名称	权限	复位值	描述
31:0	SoftLBReq	r/w	0	Software last burst request

### 7.5.12 DMA\_SoftLSReq

地址: 0x3000102c

SoftLSReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftLSReq

位	名称	权限	复位值	描述
31:0	SoftLSReq	r/w	0	Software last single request

### 7.5.13 DMA\_Config

地址: 0x30001030

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD M E

位	名称	权限	复位值	描述
31:2	RSVD			
1	M	r/w	0	AHB Master endianness configuration: 0 = little-endian, 1 = big-endian
0	E	r/w	0	SMDMA Enable.

### 7.5.14 DMA\_Sync

地址: 0x30001034

DMA\_Sync

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DMA\_Sync

位	名称	权限	复位值	描述
31:0	DMA_Sync	r/w	0	DMA synchronization logic for DMA request signals: 0 = enable, 1 = disable

### 7.5.15 DMA\_C0SrcAddr

地址: 0x30001100

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	DMA source address

### 7.5.16 DMA\_C0DstAddr

地址: 0x30001104

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	DMA Destination address

### 7.5.17 DMA\_C0LLI

地址: 0x30001108

LLI

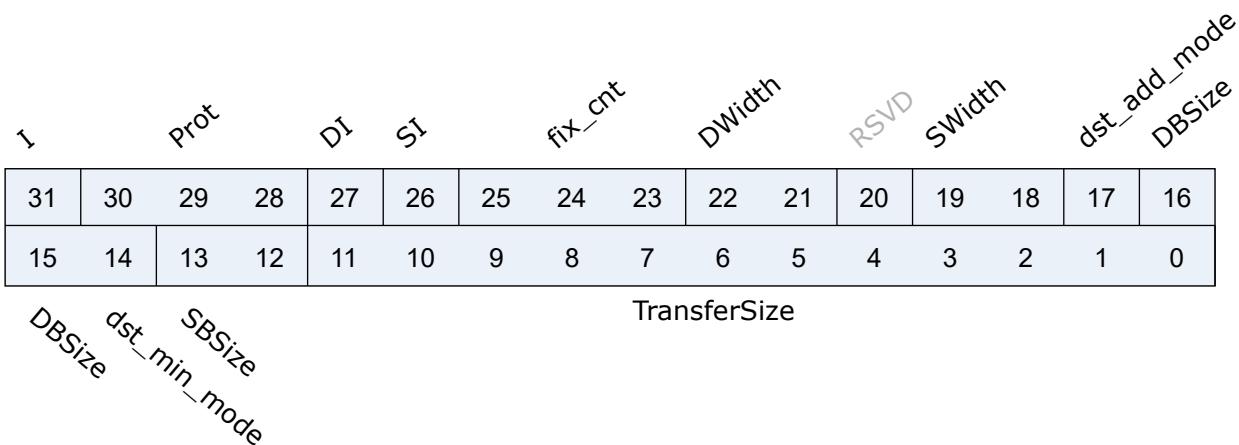
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	First linked list item. Bits [1:0] must be 0.

### 7.5.18 DMA\_C0Control

地址: 0x3000110c

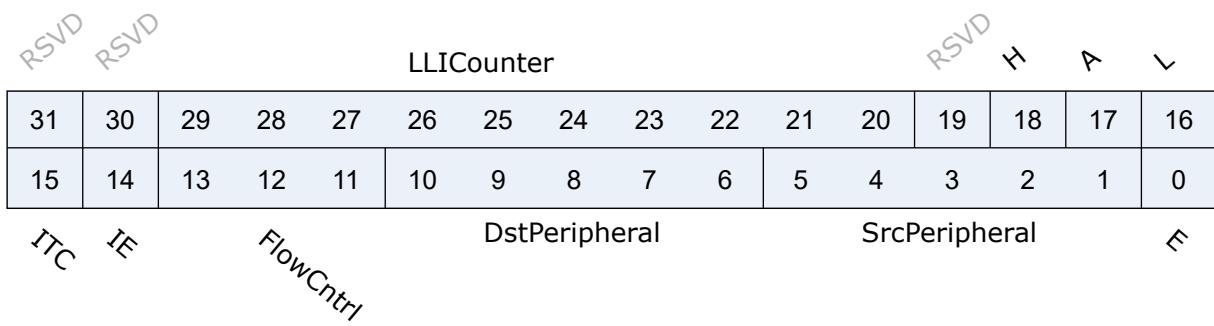


位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			

位	名称	权限	复位值	描述
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.19 DMA\_C0Config

地址: 0x30001110



位	名称	权限	复位值	描述
31:30	RSVD			
29:20	LLICounter	r	0	LLI counter. Increased 1 each LLI run. Cleared 0 when config Control.
19	RSVD			
18	H	r/w	0	Halt: 0 = enable DMA requests, 1 = ignore subsequent source DMA requests.
17	A	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
16	L	r/w	0	Lock.
15	ITC	r/w	0	Terminal count interrupt mask.
14	IE	r/w	0	Interrupt error mask.
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	E	r/w	0	Channel enable.

### 7.5.20 DMA\_C0RSVD

地址: 0x3000111c

RSVD	RSVD	RSVD	RSVD														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RSVD	DstRemnSgle	RSVD	RSVD	RSVD													
														SrcRemnSgle			

位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

### 7.5.21 DMA\_C1SrcAddr

地址: 0x30001200

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

### 7.5.22 DMA\_C1DstAddr

地址: 0x30001204

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.23 DMA\_C1LLI

地址: 0x30001208

LLI

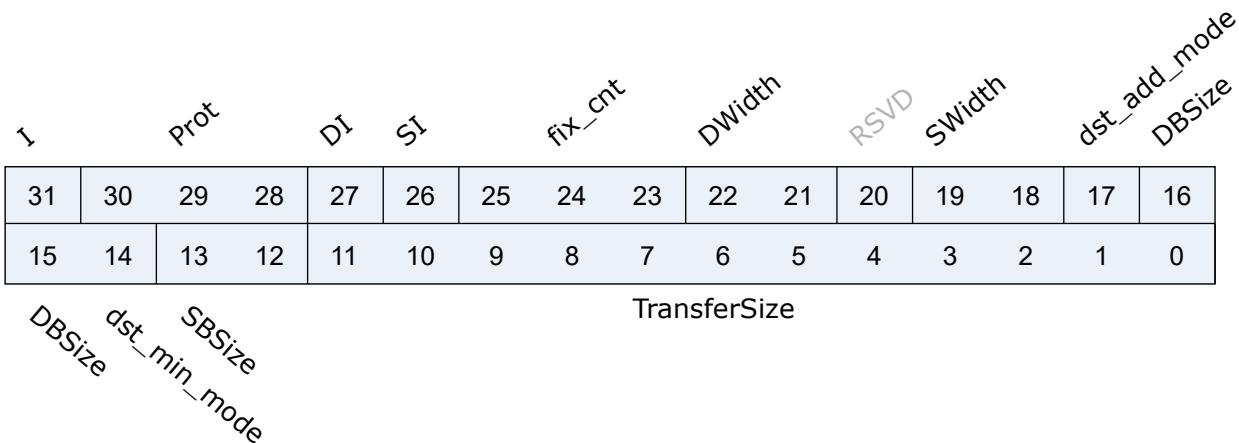
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.24 DMA\_C1Control

地址: 0x3000120c

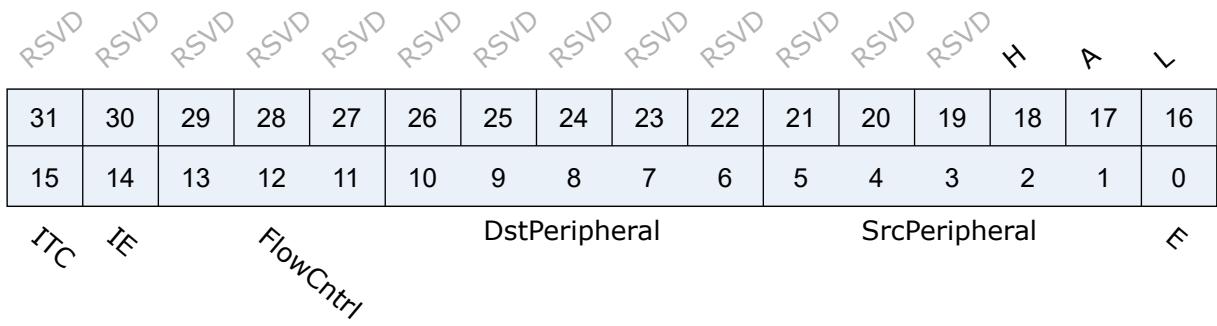


位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			

位	名称	权限	复位值	描述
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.25 DMA\_C1Config

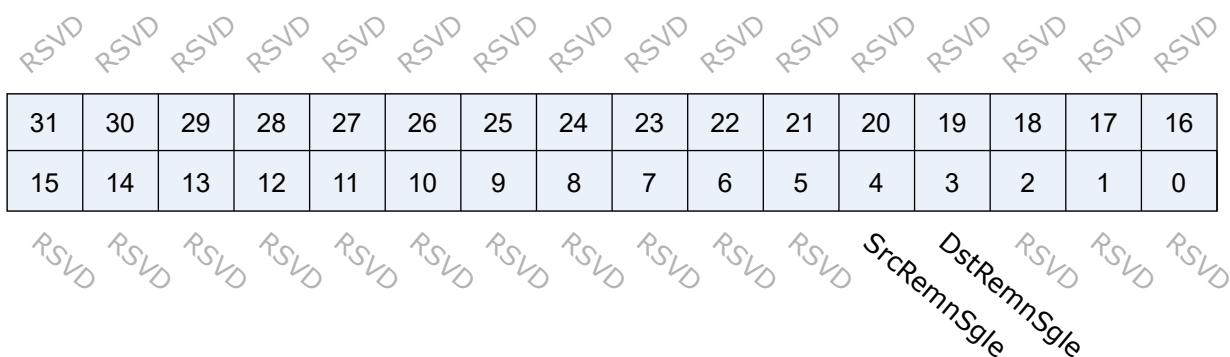
地址: 0x30001210



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

### 7.5.26 DMA\_C1RSVD

地址: 0x3000121c



位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

### 7.5.27 DMA\_C2SrcAddr

地址: 0x30001300

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

### 7.5.28 DMA\_C2DstAddr

地址: 0x30001304

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.29 DMA\_C2LLI

地址: 0x30001308

LLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.30 DMA\_C2Control

地址: 0x3000130c

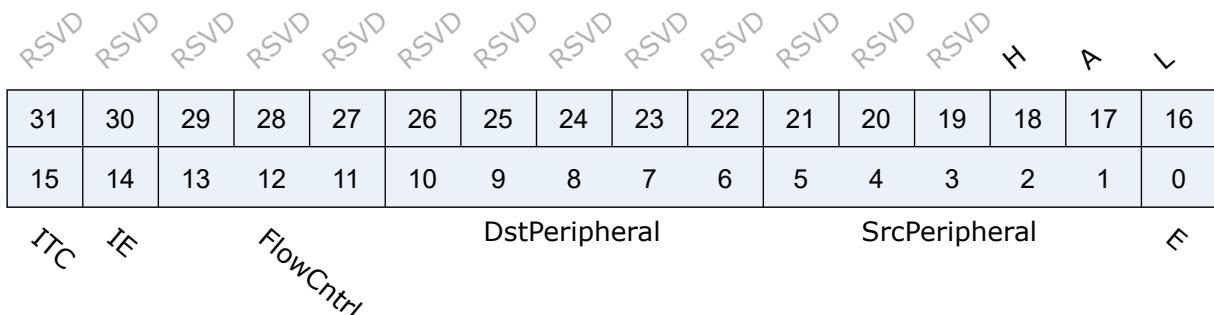
1	Prot	DI	SI	fix_cnt	DWidth	RSVD	SWidth	dst_add_mode	DBSize
31	30	29	28	27	26	25	24	23	22
15	14	13	12	11	10	9	8	7	6
									TransferSize
									DBSize
									dst_min_mode
									SBSIZE

位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*SWidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic

位	名称	权限	复位值	描述
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.31 DMA\_C2Config

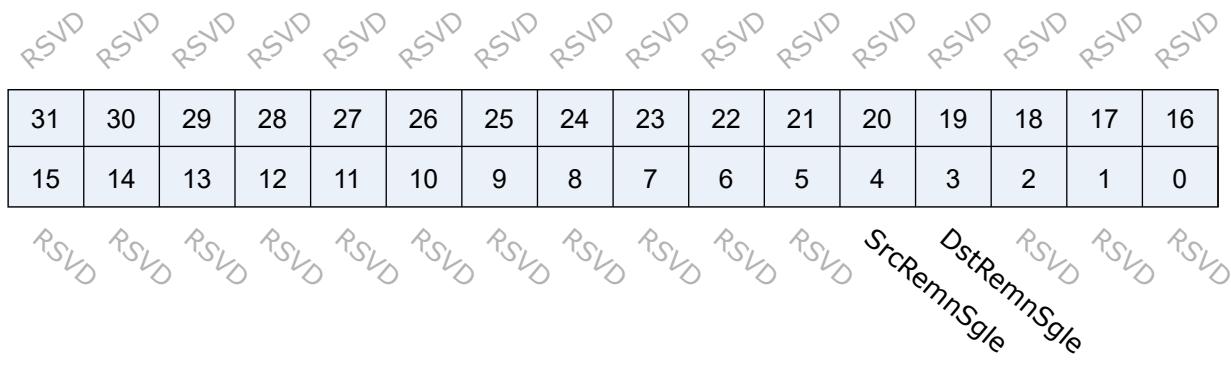
地址: 0x30001310



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

### 7.5.32 DMA\_C2RSVD

地址: 0x3000131c



位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

### 7.5.33 DMA\_C3SrcAddr

地址: 0x30001400

SrcAddr															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

### 7.5.34 DMA\_C3DstAddr

地址: 0x30001404

DstAddr															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.35 DMA\_C3LLI

地址: 0x30001408

LLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.36 DMA\_C3Control

地址: 0x3000140c

1	Prot	DI	SI	fix_cnt	DWidth	RSVD	SWidth	dst_add_mode	DBSize
31	30	29	28	27	26	25	24	23	22
15	14	13	12	11	10	9	8	7	6

TransferSize

DBSize

dst\_min\_mode

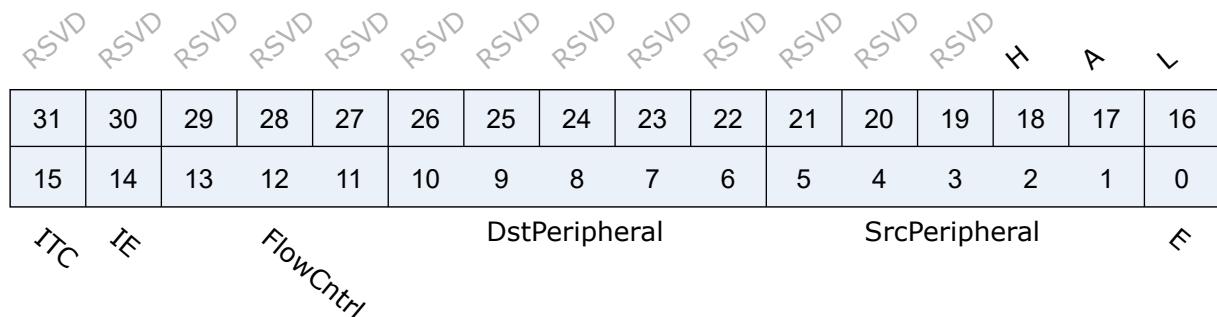
SBSIZE

位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.

位	名称	权限	复位值	描述
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.37 DMA\_C3Config

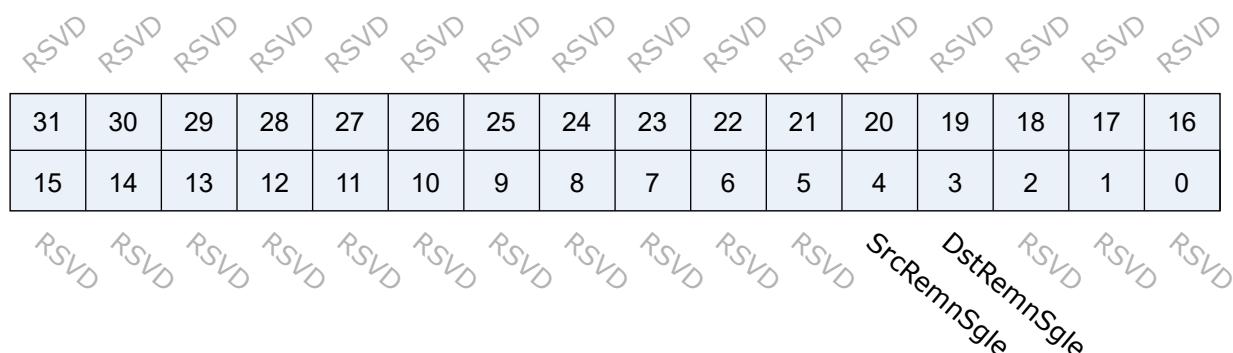
地址: 0x30001310



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

### 7.5.38 DMA\_C3RSVD

地址: 0x3000131c



位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

### 7.5.39 DMA\_C4SrcAddr

地址: 0x30001500

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

### 7.5.40 DMA\_C4DstAddr

地址: 0x30001504

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.41 DMA\_C4LLI

地址: 0x30001508

LLI

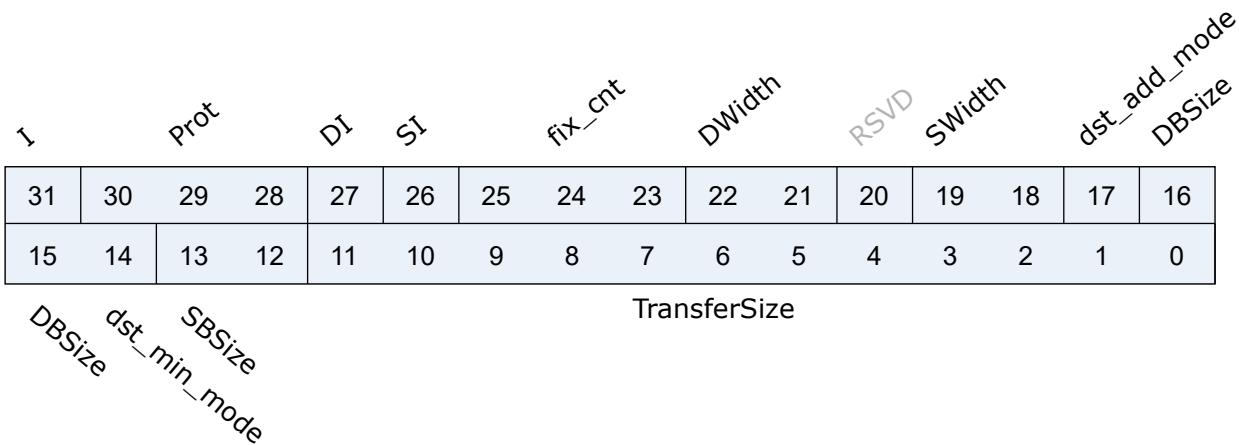
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.42 DMA\_C4Control

地址: 0x3000150c

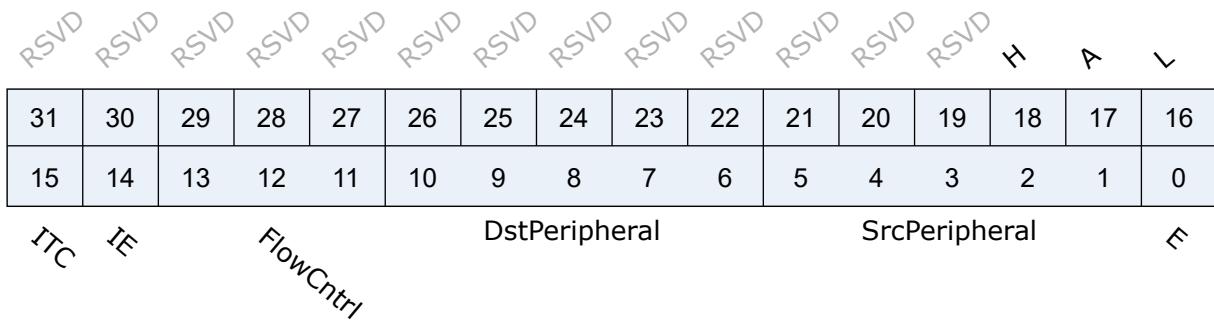


位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			

位	名称	权限	复位值	描述
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.43 DMA\_C4Config

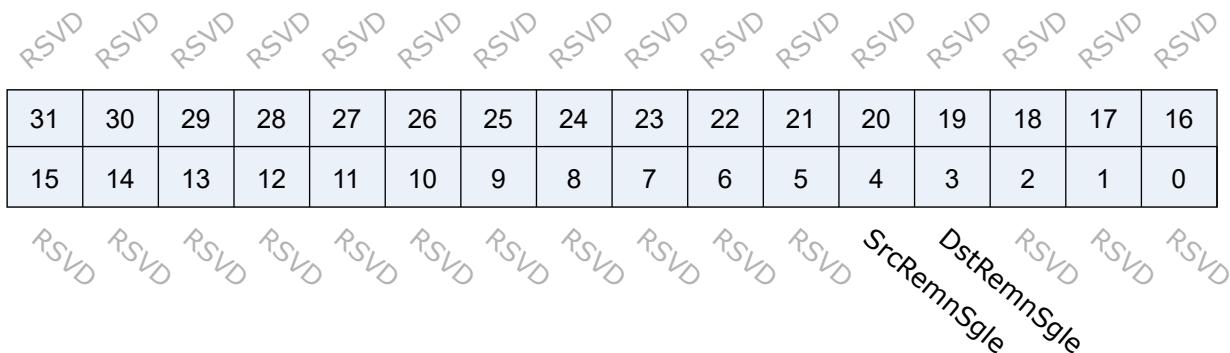
地址: 0x30001510



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

### 7.5.44 DMA\_C4RSVD

地址: 0x3000151c



位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSngle	r/w	0	Source remain single issue mode
3	DstRemnSngle	r/w	0	Destination remain single issue mode
2:0	RSVD			

### 7.5.45 DMA\_C5SrcAddr

地址: 0x30001600

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

### 7.5.46 DMA\_C5DstAddr

地址: 0x30001604

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.47 DMA\_C5LLI

地址: 0x30001608

LLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.48 DMA\_C5Control

地址: 0x3000160c

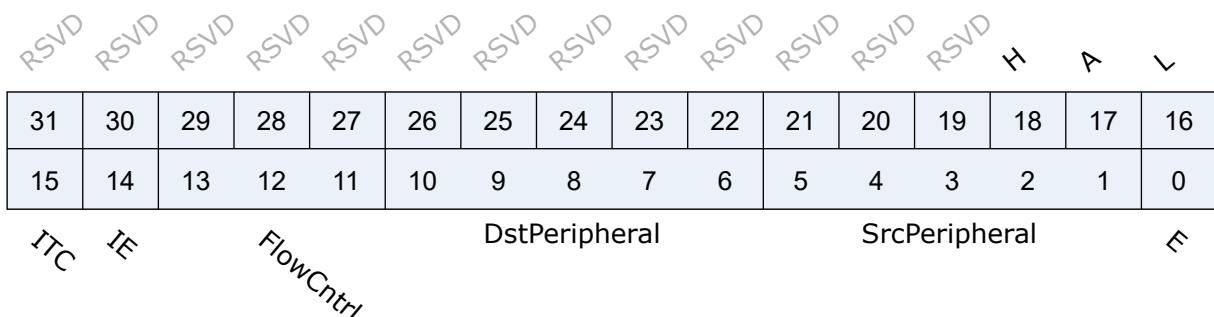
1	Prot	DI	SI	fix_cnt	DWidth	RSVD	SWidth	dst_add_mode	DBSize
31	30	29	28	27	26	25	24	23	22
15	14	13	12	11	10	9	8	7	6
									TransferSize
									DBSize
									dst_min_mode
									SBSIZE

位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*SWidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic

位	名称	权限	复位值	描述
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.49 DMA\_C5Config

地址: 0x30001610



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

## 7.5.50 DMA\_C5RSVD

地址: 0x3000161c

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD   RSVD

SrcRemnSgle   DstRemnSgle

位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

## 7.5.51 DMA\_C6SrcAddr

地址: 0x30001700

SrcAddr															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

## 7.5.52 DMA\_C6DstAddr

地址: 0x30001704

DstAddr															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.53 DMA\_C6LLI

地址: 0x30001708

LLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.54 DMA\_C6Control

地址: 0x3000170c

1	Prot	DI	SI	fix_cnt	DWidth	RSVD	SWidth	dst_add_mode	DBSize
31	30	29	28	27	26	25	24	23	22
15	14	13	12	11	10	9	8	7	6

TransferSize

DBSize

dst\_min\_mode

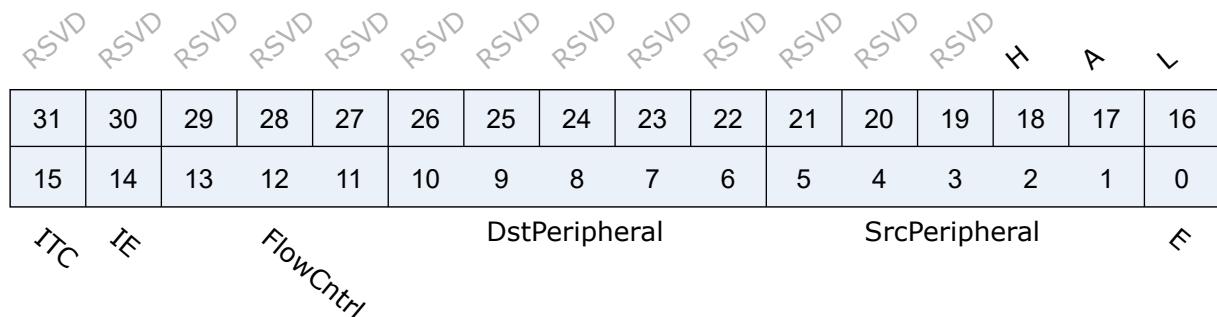
SBSIZE

位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.

位	名称	权限	复位值	描述
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.55 DMA\_C6Config

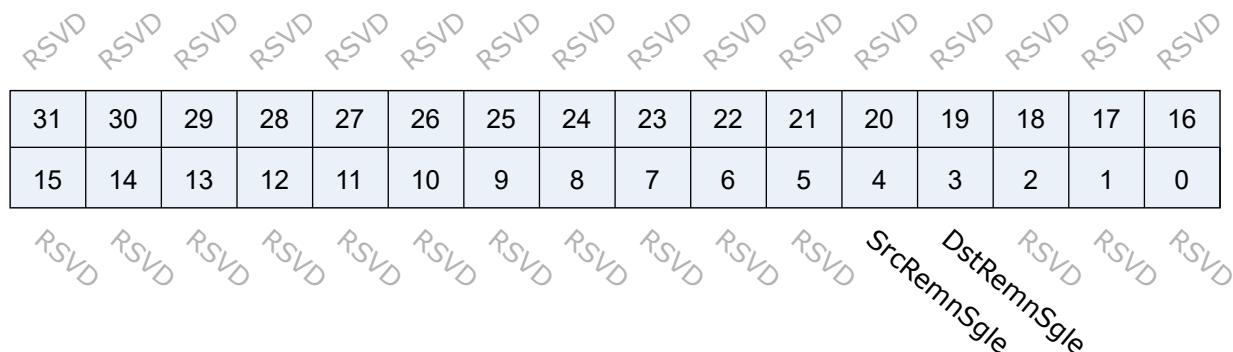
地址: 0x30001710



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

### 7.5.56 DMA\_C6RSVD

地址: 0x3000171c



位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

### 7.5.57 DMA\_C7SrcAddr

地址: 0x30001800

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	

### 7.5.58 DMA\_C7DstAddr

地址: 0x30001804

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	

### 7.5.59 DMA\_C7LLI

地址: 0x30001808

LLI

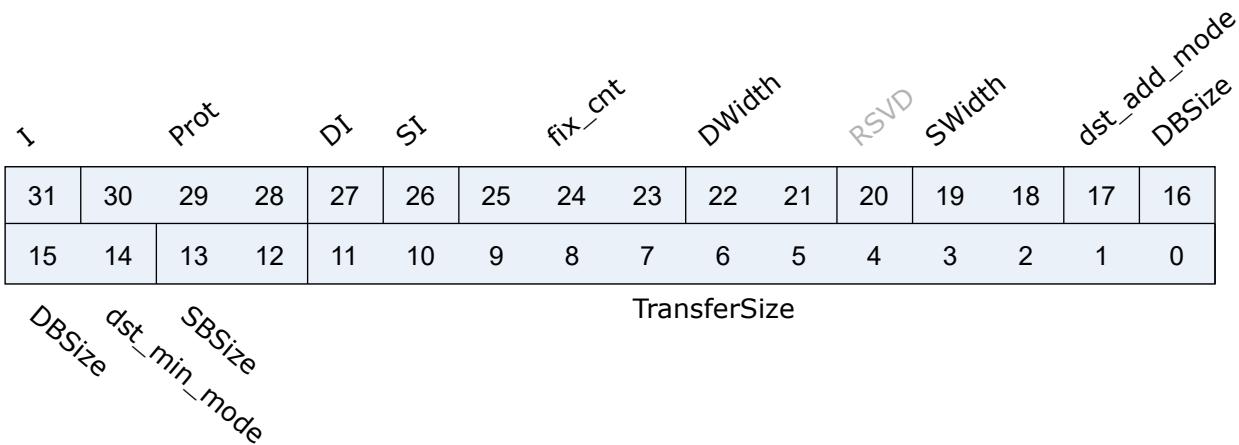
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

位	名称	权限	复位值	描述
31:0	LLI	r/w	0	

### 7.5.60 DMA\_C7Control

地址: 0x3000180c

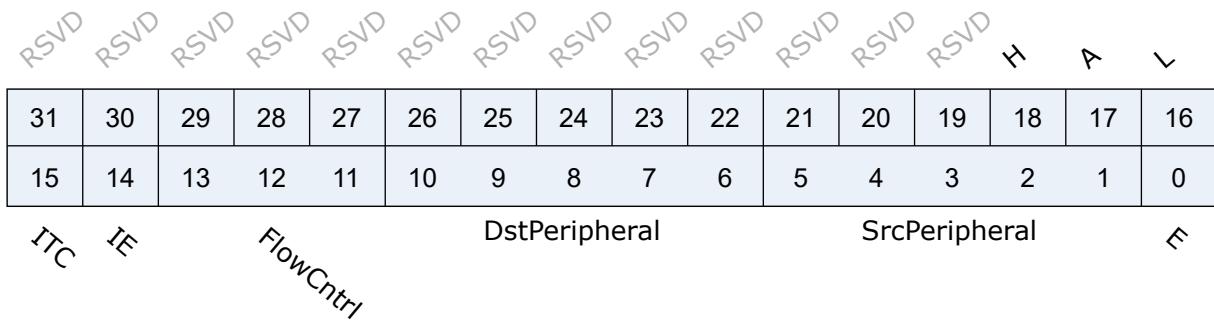


位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			

位	名称	权限	复位值	描述
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

### 7.5.61 DMA\_C7Config

地址: 0x30001810



位	名称	权限	复位值	描述
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

### 7.5.62 DMA\_C7RSVD

地址: 0x3000181c

RSVD	RSVD	RSVD	RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
																SrcRemnSgle	DstRemnSgle		

位	名称	权限	复位值	描述
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

## 8.1 简介

DMA2D 即 2D DMA(2 Dimensional Direct Memory Access) 是一种在 1D DMA 的基础上进行扩展，从而使得 2D DMA 控制器可以按照设定的步幅搬运数据的技术(关于 1D DMA 的介绍详见 DMA 章节)。数据的内存地址可以是非连续的，按照一定的间隔递增、递减或不变。2D DMA 控制器有 2 组独立专用通道，管理外围设备和内存之间的数据传输以提高总线效率。主要有三种类型传输包括内存至内存、内存至外设、外设至内存。并支持 LLI 链接列表功能。

## 8.2 主要特征

- 2 组独立专用通道
- 可配置为 1D DMA 或 2D DMA 功能
- 独立控制来源与目标存取宽度(单字节、双字节、四字节)
- 每个通道独立作为读写缓存
- 每个通道可被独立的外设硬件触发或是软件触发
- 支持外设包括 UART、I2C、SPI、DBI、DSI。
- 八种流程控制
  - DMA 流程控制，来源内存、目标内存
  - DMA 流程控制，来源内存、目标外设
  - DMA 流程控制，来源外设、目标内存
  - DMA 流程控制，来源外设、目标外设
  - 目标外设流程控制，来源外设、目标外设
  - 目标外设流程控制，来源内存、目标外设
  - 来源外设流程控制，来源外设、目标内存
  - 来源外设流程控制，来源外设、目标外设
- 支持 LLI 链表功能，提高 DMA 效率，每个节点可单独设置是否触发中断
- 以下特征为 2D DMA 特有：
  - 支持 8/16/32-bit 像素图形在任意方向上的平移
  - 支持 8/16/32-bit 像素图形 90/180/270 度旋转

- 支持 8/16/32-bit 像素图形水平/垂直翻折
- 支持 8/16/32-bit 像素图形填充
- 支持 8/16/24/32-bit Color Key

## 8.3 功能描述

### 8.3.1 工作原理

2D DMA 在 1D DMA 的基础上，增加了源地址的 X 步幅值 SRC XINCR、X 循环值 SRC XCOUNT、Y 步幅值 SRC YINCR 和 Y 循环值 SRC YCOUNT，以及目的地址的 X 步幅值 DEST XINCR、X 循环值 DEST XCOUNT 和 Y 步幅值 DEST YINCR 配置 (DEST YCOUNT 不需要配置，由其他值计算得到)，其中步幅值可为负数。2D DMA 传输可看作是两层循环，外层循环为循环 YCOUNT 次，每次循环地址递增 YINCR 字节 (YINCR 为负数时递减)；内层循环为循环 XCOUNT 次，每次循环地址递增 XINCR 字节 (XINCR 为负数时递减)。工作原理如下图所示：

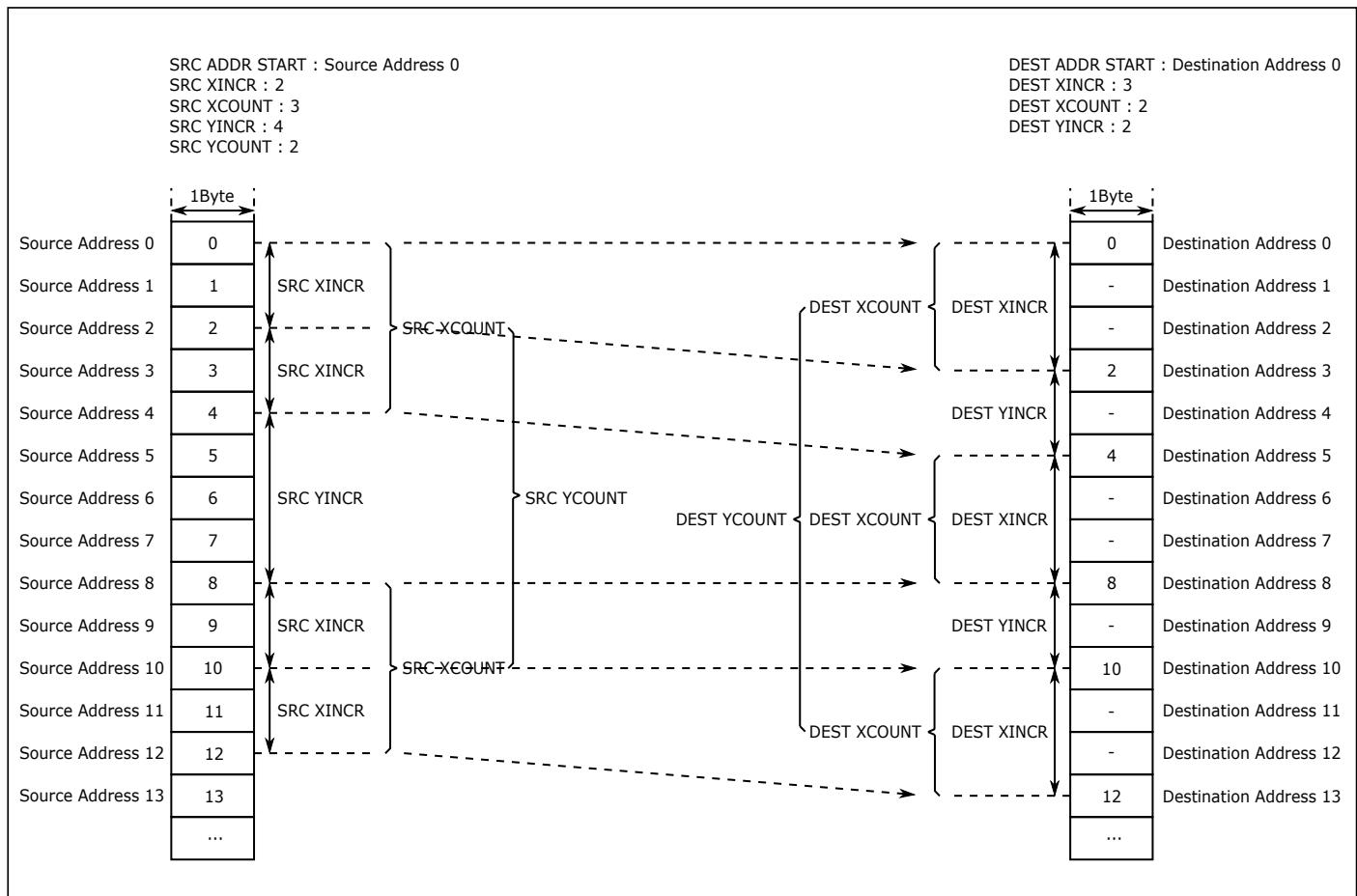


图 8.1: 工作原理

### 8.3.2 图像平移

将内存数据看成是一张矩形图像，通过步幅值和循环值的设置，可以实现矩形图像平移的效果，如下图所示：

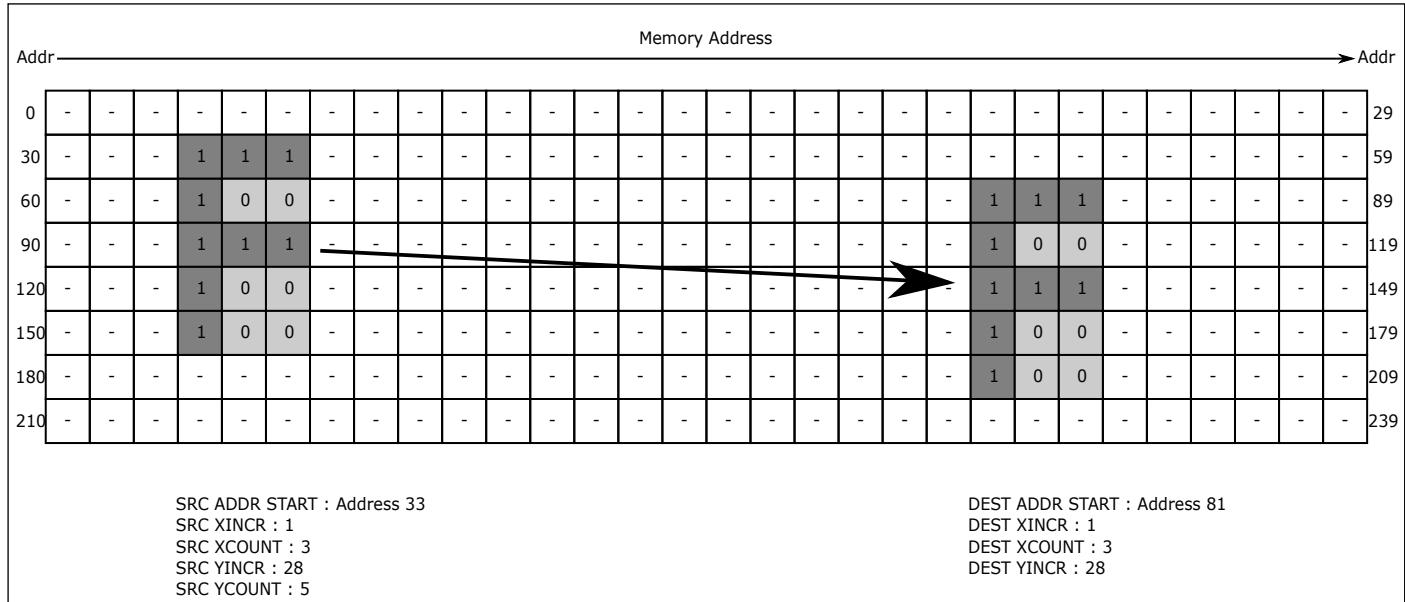


图 8.2: 图像平移

按图中示例，假设一个像素点宽 (P) 为 8-bit, 2D DMA 的 transfer width 设置为 8-bit, 在一张大小为 M\*N(图中为 30\*8) 的图像中将一个大小为 W\*H(图中为 3\*5) 的矩形区域从坐标 (x1,y1)(图中为 (3,1)) 平移到坐标 (x2,y2)(图中为 (21,2)), 且图像数据起始地址从地址 addr0(图中为 0) 开始，则 2D DMA 的相关参数计算过程如下：

- 源地址 SRC ADDR START =  $addr0 + (M \cdot y1 + x1) \cdot P = 0 + (30 \cdot 1 + 3) \cdot 1 = 33$
- 源地址 X 步幅值 SRC XINCR =  $P = 1$
- 源地址 X 循环值 SRC XCOUNT =  $W = 3$
- 源地址 Y 步幅值 SRC YINCR =  $(M - (W - 1)) \cdot P = (30 - (3 - 1)) \cdot 1 = 28$
- 源地址 Y 循环值 SRC YCOUNT =  $H = 5$
- 目的地址 DEST ADDR START =  $addr0 + (M \cdot y2 + x2) \cdot P = 0 + (30 \cdot 2 + 21) \cdot 1 = 81$
- 目的地址 X 步幅值 DEST XINCR =  $P = 1$
- 目的地址 X 循环值 DEST XCOUNT =  $W = 3$
- 目的地址 Y 步幅值 DEST YINCR =  $(M - (W - 1)) \cdot P = (30 - (3 - 1)) \cdot 1 = 28$

### 8.3.3 图像旋转

将内存数据看成是一张矩形图像，通过步幅值和循环值的设置，可以实现矩形图像 90/180/270 度旋转的效果，如下图所示：

		Memory Address																																
Addr																												Addr						
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	0	-	29	
30	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	0	-	59		
60	-	-	-	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	-	89		
90	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	119
120	-	-	-	1	0	0	-	-	-	-	-	-	-	-	-	-	-	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	149
150	-	-	-	1	0	0	-	-	-	-	1	1	1	1	1	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	179	
180	-	-	-	-	-	-	-	-	-	0	0	1	0	1	-	-	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	209	
210	-	-	-	-	-	-	-	-	-	0	0	1	0	1	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	239	
		SRC ADDR START : Address 33 SRC XINCR : 1 SRC XCOUNT : 3 SRC YINCR : 28 SRC YCOUNT : 5												DEST ADDR START : Address 164 DEST XINCR : 30 DEST XCOUNT : 3 DEST YINCR : -61												DEST ADDR START : Address 229 DEST XINCR : -1 DEST XCOUNT : 3 DEST YINCR : 61								

图 8.3: 图像旋转

按图中示例，假设一个像素点宽 (P) 为 8-bit, 2D DMA 的 transfer width 设置为 8-bit, 在一张大小为 M\*N(图中为 30\*8) 的图像中将一个大小为 W\*H(图中为 3\*5) 的矩形区域从坐标 (x1,y1)(图中为 (3,1)) 顺时针旋转 90 度到坐标 (x2,y2)(图中为 (14,5)), 且图像数据起始地址从地址 addr0(图中为 0) 开始，则 2D DMA 的相关参数计算过程如下：

- 源地址 SRC ADDR START =  $addr0 + (M * y1 + x1) * P = 0 + (30 * 1 + 3) * 1 = 33$
- 源地址 X 步幅值 SRC XINCR =  $P = 1$
- 源地址 X 循环值 SRC XCOUNT =  $W = 3$
- 源地址 Y 步幅值 SRC YINCR =  $(M - (W - 1)) * P = (30 - (3 - 1)) * 1 = 28$
- 源地址 Y 循环值 SRC YCOUNT =  $H = 5$
- 目的地址 DEST ADDR START =  $addr0 + (M * y2 + x2) * P = 0 + (30 * 5 + 14) * 1 = 164$
- 目的地址 X 步幅值 DEST XINCR =  $M * P = 30 * 1 = 30$
- 目的地址 X 循环值 DEST XCOUNT =  $W = 3$
- 目的地址 Y 步幅值 DEST YINCR =  $-(M * (W - 1) + 1) * P = -(30 * (3 - 1) + 1) * 1 = -61$

将该矩形区域从坐标 (x1,y1)(图中为 (3,1)) 顺时针旋转 180 度到坐标 (x3,y3)(图中为 (19,7)) 的相关参数计算过程如下：

- 源地址 SRC ADDR START =  $addr0 + (M * y1 + x1) * P = 0 + (30 * 1 + 3) * 1 = 33$
- 源地址 X 步幅值 SRC XINCR =  $P = 1$
- 源地址 X 循环值 SRC XCOUNT =  $W = 3$
- 源地址 Y 步幅值 SRC YINCR =  $(M - (W - 1)) * P = (30 - (3 - 1)) * 1 = 28$
- 源地址 Y 循环值 SRC YCOUNT =  $H = 5$
- 目的地址 DEST ADDR START =  $addr0 + (M * y3 + x3) * P = 0 + (30 * 7 + 19) * 1 = 229$
- 目的地址 X 步幅值 DEST XINCR =  $-P = -1$
- 目的地址 X 循环值 DEST XCOUNT =  $W = 3$

- 目的地址 Y 步幅值 DEST YINCR = -(M-(W-1))\*P = -(30-(3-1))\*1 = -28

将该矩形区域从坐标 (x1,y1)(图中为 (3,1)) 顺时针旋转 270 度到坐标 (x4,y4)(图中为 (24,2)) 的相关参数计算过程如下：

- 源地址 SRC ADDR START = addr0+(M\*y1+x1)\*P = 0+(30\*1+3)\*1 = 33
- 源地址 X 步幅值 SRC XINCR = P = 1
- 源地址 X 循环值 SRC XCOUNT = W = 3
- 源地址 Y 步幅值 SRC YINCR = (M-(W-1))\*P = (30-(3-1))\*1 = 28
- 源地址 Y 循环值 SRC YCOUNT = H = 5
- 目的地址 DEST ADDR START = addr0+(M\*y4+x4)\*P = 0+(30\*2+24)\*1 = 84
- 目的地址 X 步幅值 DEST XINCR = -M\*P = -30\*1 = -30
- 目的地址 X 循环值 DEST XCOUNT = W = 3
- 目的地址 Y 步幅值 DEST YINCR = (M\*(W-1)+1)\*P = (30\*(3-1)+1)\*1 = 61

### 8.3.4 图像翻折

将内存数据看成是一张矩形图像，通过步幅值和循环值的设置，可以实现矩形图像水平/垂直翻折的效果，如下图所示：

Memory Address																														
Addr	Addr																													
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29
30	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	59
60	-	-	-	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	89
90	-	-	-	1	1	1	-	-	-	-	-	-	-	1	1	1	-	-	-	-	-	1	0	0	-	-	-	-	-	119
120	-	-	-	1	0	0	-	-	-	-	-	-	-	0	0	1	-	-	-	-	-	1	1	1	-	-	-	-	-	149
150	-	-	-	1	0	0	-	-	-	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	179	
180	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	209
210	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	239
SRC ADDR START : Address 33										DEST ADDR START : Address 105										DEST ADDR START : Address 142										
SRC XINCR : 1										DEST XINCR : -1										DEST XINCR : 1										
SRC XCOUNT : 3										DEST XCOUNT : 3										DEST XCOUNT : 3										
SRC YINCR : 28										DEST YINCR : 32										DEST YINCR : -32										
SRC YCOUNT : 5																														

图 8.4: 图像翻折

按图中示例，假设一个像素点宽 (P) 为 8-bit, 2D DMA 的 transfer width 设置为 8-bit，在一张大小为 M\*N(图中为 30\*8) 的图像中将一个大小为 W\*H(图中为 3\*5) 的矩形区域从坐标 (x1,y1)(图中为 (3,1)) 水平翻折到坐标 (x2,y2)(图中为 (15,3)), 且图像数据起始地址从地址 addr0(图中为 0) 开始，则 2D DMA 的相关参数计算过程如下：

- 源地址 SRC ADDR START = addr0+(M\*y1+x1)\*P = 0+(30\*1+3)\*1 = 33
- 源地址 X 步幅值 SRC XINCR = P = 1
- 源地址 X 循环值 SRC XCOUNT = W = 3
- 源地址 Y 步幅值 SRC YINCR = (M-(W-1))\*P = (30-(3-1))\*1 = 28

- 源地址 Y 循环值 SRC YCOUNT = H = 5
  - 目的地址 DEST ADDR START = addr0+(M\*y2+x2)\*P = 0+(30\*3+15)\*1 = 105
  - 目的地址 X 步幅值 DEST XINCR = -P = -1
  - 目的地址 X 循环值 DEST XCOUNT = W = 3
  - 目的地址 Y 步幅值 DEST YINCR = (M+(W-1))\*P = (30+(3-1))\*1 = 32

将该矩形区域从坐标  $(x_1, y_1)$  (图中为  $(3, 1)$ ) 垂直翻折到坐标  $(x_3, y_3)$  (图中为  $(22, 4)$ ) 的相关参数计算过程如下:

- 源地址 SRC ADDR START =  $addr0 + (M^*y1+x1)^*P = 0 + (30^*1+3)^*1 = 33$
  - 源地址 X 步幅值 SRC XINCR = P = 1
  - 源地址 X 循环值 SRC XCOUNT = W = 3
  - 源地址 Y 步幅值 SRC YINCR =  $(M-(W-1))^*P = (30-(3-1))^*1 = 28$
  - 源地址 Y 循环值 SRC YCOUNT = H = 5
  - 目的地址 DEST ADDR START =  $addr0 + (M^*y3+x3)^*P = 0 + (30^*4+22)^*1 = 142$
  - 目的地址 X 步幅值 DEST XINCR = P = 1
  - 目的地址 X 循环值 DEST XCOUNT = W = 3
  - 目的地址 Y 步幅值 DEST YINCR =  $-(M+(W-1))^*P = -(30+(3-1))^*1 = -32$

### 8.3.5 图像填充

将内存数据看成是一张矩形图像，通过步幅值和循环值的设置，可以实现矩形图像填充的效果，如下图所示：

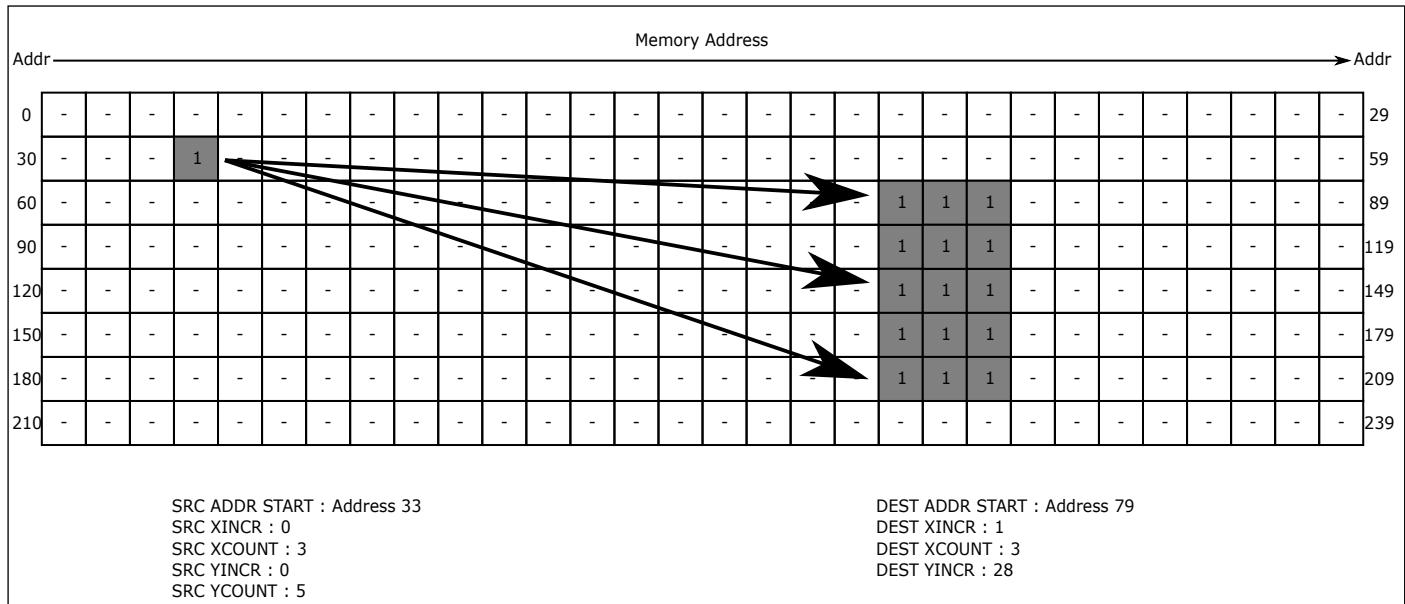


图 8.5: 图像填充

按图中示例，假设一个像素点宽 (P) 为 8-bit, 2D DMA 的 transfer width 设置为 8-bit, 在一张大小为  $M \times N$ (图中为  $30 \times 8$ ) 的图像中将一个大小为  $W \times H$ (图中为  $3 \times 5$ )、起始坐标为  $(x2, y2)$ (图中为  $(19, 2)$ ) 的矩形区域用坐标  $(x1, y1)$ (图中为  $(3, 1)$ ) 中的值填充，且图像数据起始地址从地址  $addr0$ (图中为  $0$ ) 开始，则 2D DMA 的相关参数计算过程如下：

- 源地址 SRC ADDR START = addr0+(M\*y1+x1)\*P = 0+(30\*1+3)\*1 = 33
- 源地址 X 步幅值 SRC XINCR = 0
- 源地址 X 循环值 SRC XCOUNT = W = 3
- 源地址 Y 步幅值 SRC YINCR = 0
- 源地址 Y 循环值 SRC YCOUNT = H = 5
- 目的地址 DEST ADDR START = addr0+(M\*y2+x2)\*P = 0+(30\*2+19)\*1 = 79
- 目的地址 X 步幅值 DEST XINCR = P = 1
- 目的地址 X 循环值 DEST XCOUNT = W = 3
- 目的地址 Y 步幅值 DEST YINCR = (M-(W-1))\*P = (30-(3-1))\*1 = 28

### 8.3.6 Color Key

当开启 Color Key 功能时，2D DMA 在搬运数据的过程中如果遇到和设置的 Color Key 值相等的值时，会跳过该值不做搬运，被跳过的地址里存储的还是原来的值。Color Key 的宽度可设置为 8/16/24/32 位宽，工作原理如下图所示：

Memory Address																															
Addr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Addr	
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29	
30	-	-	-	0xFF	0xFF	0xFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	59		
60	-	-	-	0xFF	0x55	0x55	-	-	-	-	-	-	-	-	-	-	-	-	0xFF	0xFF	0xFF	-	-	-	-	-	-	-	89		
90	-	-	-	0xFF	0xFF	0xFF	-	-	-	-	-	-	-	-	-	-	-	-	0xFF	-	-	-	-	-	-	-	-	-	119		
120	-	-	-	0xFF	0x55	0x55	-	-	-	-	-	-	-	-	-	-	-	-	0xFF	0xFF	0xFF	-	-	-	-	-	-	-	149		
150	-	-	-	0xFF	0x55	0x55	-	-	-	-	-	-	-	-	-	-	-	-	0xFF	-	-	-	-	-	-	-	-	-	179		
180	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0xFF	-	-	-	-	-	-	-	-	-	209		
210	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	239	
SRC ADDR START : Address 33																Color Key Width : 8-bit				DEST ADDR START : Address 79											
SRC XINCR : 1																Color Key Value : 0x55				DEST XINCR : 1				DEST XCOUNT : 3				DEST YINCR : 28			
SRC YINCR : 28																SRC YCOUNT : 5															

图 8.6: Color Key

按图中示例，Color Key 位宽设置为 8-bit，Key 值设置为 0x55，2D DMA 在搬运包含“F”图像的矩形区域时，遇到 0x55 值会跳过不做搬运，可以看到结果中只搬运了矩形区域的“F”，值为 0x55 的背景部分没有搬运。

## 8.4 寄存器描述

名称	描述
DMA2D_IntStatus	
DMA2D_IntTCStatus	



名称	描述
DMA2D_IntTCClear	
DMA2D_EnbldChns	
DMA2D_Config	
DMA2D_Sync	
DMA2D_SoftBReq	
DMA2D_SoftLBReq	
DMA2D_SoftSReq	
DMA2D_SoftLSReq	
DMA2D_C0SrcAddr	
DMA2D_C0DstAddr	
DMA2D_C0LLI	
DMA2D_C0_BUS	
DMA2D_C0_SRC_CNT	
DMA2D_C0_SRC_XIC	
DMA2D_C0_SRC_YIC	
DMA2D_C0_DST_CNT	
DMA2D_C0_DST_XIC	
DMA2D_C0_DST_YIC	
DMA2D_C0_KEY	
DMA2D_C0_KEY_EN	
DMA2D_C0_CFG	

## 8.4.1 DMA2D\_IntStatus

地址: 0x30006000

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntStatus	r	0	Status of the DMA interrupts after masking

#### 8.4.2 DMA2D\_IntTCStatus

地址: 0x30006004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
IntTCStatus															

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntTCStatus	r	0	Interrupt terminal count request status

#### 8.4.3 DMA2D\_IntTCClear

地址: 0x30006008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
IntTCClear															

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	IntTCClear	w1p	0	Terminal count request clear

#### 8.4.4 DMA2D\_EnBldChns

地址: 0x3000600c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

EnabledChannels

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	EnabledChannels	r	0	Channel enable status

#### 8.4.5 DMA2D\_Config

地址: 0x30006010

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	E

位	名称	权限	复位值	描述
31:24	RSVD			
23:16	ChClkCG	r/w	8'hff	Channel Clock cg enable
15:1	RSVD			
0	E	r/w	0	SMDMA Enable.

#### 8.4.6 DMA2D\_Sync

地址: 0x30006014

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

DMA\_Sync

位	名称	权限	复位值	描述
31:0	DMA_Sync	r/w	0	DMA synchronization logic for DMA request signals: 0 = enable, 1 = disable

#### 8.4.7 DMA2D\_SoftBReq

地址: 0x30006018

SoftBReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftBReq

位	名称	权限	复位值	描述
31:0	SoftBReq	r/w	0	Software burst request

#### 8.4.8 DMA2D\_SoftLBReq

地址: 0x3000601c

SoftLBReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftLBReq

位	名称	权限	复位值	描述
31:0	SoftLBReq	r/w	0	Software last burst request

#### 8.4.9 DMA2D\_SoftSReq

地址: 0x30006020

SoftSReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftSReq

位	名称	权限	复位值	描述
31:0	SoftSReq	r/w	0	Software signle request

#### 8.4.10 DMA2D\_SoftLSReq

地址: 0x30006024

SoftLSReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftLSReq

位	名称	权限	复位值	描述
31:0	SoftLSReq	r/w	0	Software last signle request

#### 8.4.11 DMA2D\_C0SrcAddr

地址: 0x30006100

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

位	名称	权限	复位值	描述
31:0	SrcAddr	r/w	0	DMA source address

#### 8.4.12 DMA2D\_C0DstAddr

地址: 0x30006104

DstAddr

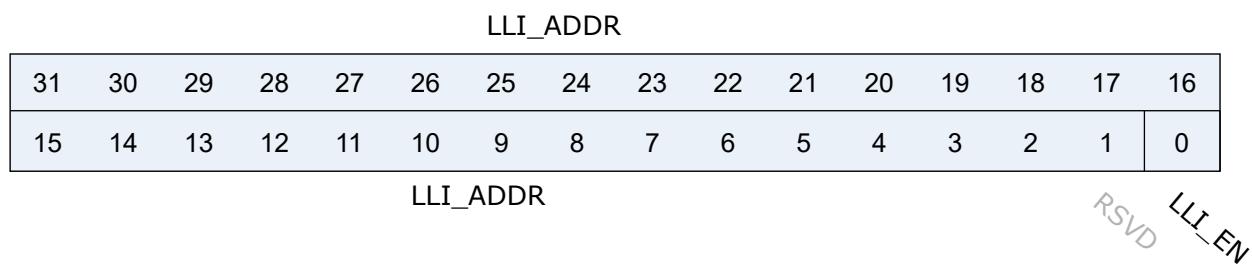
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

位	名称	权限	复位值	描述
31:0	DstAddr	r/w	0	DMA Destination address

### 8.4.13 DMA2D\_C0LLI

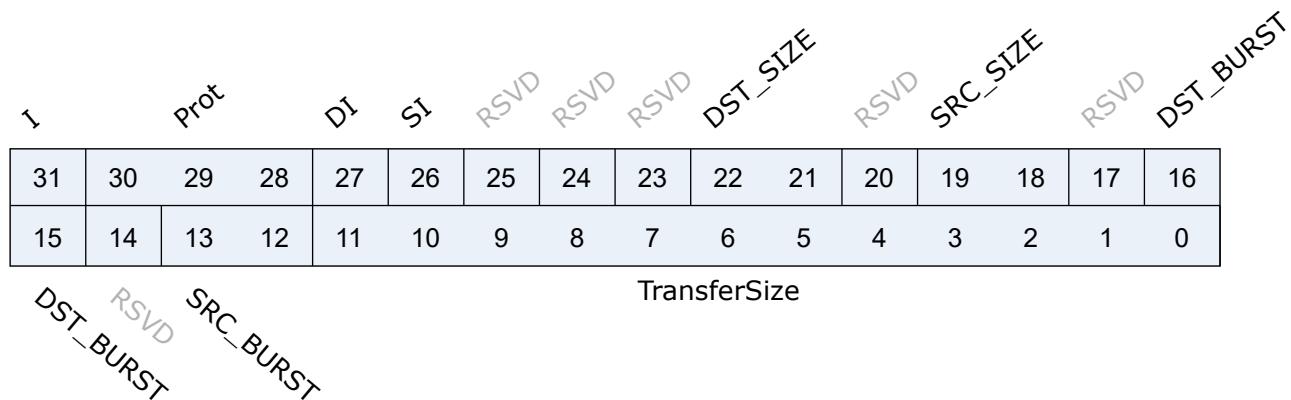
地址: 0x30006108



位	名称	权限	复位值	描述
31:2	LLI_ADDR	r/w	0	Next linked list address
1	RSVD			
0	LLI_EN	r/w	0	Next LLI enable

### 8.4.14 DMA2D\_C0\_BUS

地址: 0x3000610c



位	名称	权限	复位值	描述
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	Hprot Setting
27	DI	r/w	1	Only work when reg_dma_2d_en = 0 Destination increment. When set, the Destination address is incremented after each transfer.

位	名称	权限	复位值	描述
26	SI	r/w	1	Only work when reg_dma_2d_en = 0 Source increment. When set, the source address is incremented after each transfer.
25:23	RSVD			
22:21	DST_SIZE	r/w	2'd2	Destination transfer width: 8/16/32
20	RSVD			
19:18	SRC_SIZE	r/w	2'd2	Source transfer width: 8/16/32
17	RSVD			
16:15	DST_BURST	r/w	2'd1	Destination burst size: 1/4/8/16
14	RSVD			
13:12	SRC_BURST	r/w	2'd1	
11:0	TransferSize	r/w	0	Only work when reg_dma_2d_en = 0 Transfer size: 0~4095. Number of data transfers left to complete when the SMDMA is the flow controller.

#### 8.4.15 DMA2D\_C0\_SRC\_CNT

地址: 0x30006110

SRC\_Y\_CNT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SRC\_X\_CNT

位	名称	权限	复位值	描述
31:16	SRC_Y_CNT	r/w	16'd0	Only work when reg_dma_2d_en = 1 Source Y-direction Count
15:0	SRC_X_CNT	r/w	16'd0	Only work when reg_dma_2d_en = 1 Source X-direction Count

### 8.4.16 DMA2D\_C0\_SRC\_XIC

地址: 0x30006114

SRC\_X\_INCR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SRC\_X\_INCR

位	名称	权限	复位值	描述
31:0	SRC_X_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Source Y-direction address increment Must align "SWidth" setting [16] sign-bit

### 8.4.17 DMA2D\_C0\_SRC\_YIC

地址: 0x30006118

SRC\_Y\_INCR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SRC\_Y\_INCR

位	名称	权限	复位值	描述
31:0	SRC_Y_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Source Y-direction address increment Must align "SWidth" setting [16] sign-bit

### 8.4.18 DMA2D\_C0\_DST\_CNT

地址: 0x3000611c

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DST\_X\_CNT

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	DST_X_CNT	r/w	16'd0	Only work when reg_dma_2d_en = 1 Source X-direction Count

#### 8.4.19 DMA2D\_C0\_DST\_XIC

地址: 0x30006120

DST\_X\_INCR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DST\_X\_INCR

位	名称	权限	复位值	描述
31:0	DST_X_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Source Y-direction address increment Must align "SWidth" setting [16] sign-bit

#### 8.4.20 DMA2D\_C0\_DST\_YIC

地址: 0x30006124

DST\_Y\_INCR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DST\_Y\_INCR

位	名称	权限	复位值	描述
31:0	DST_Y_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Desitination Y-direction address increment Must align "DWidth" setting [16] sign-bit

### 8.4.21 DMA2D\_C0\_KEY

地址: 0x30006174

KEY3								KEY2							
KEY1								KEY0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:24	KEY3	r/w	0	Key value
23:16	KEY2	r/w	0	Key value
15:8	KEY1	r/w	0	Key value
7:0	KEY0	r/w	0	Key value

### 8.4.22 DMA2D\_C0\_KEY\_EN

地址: 0x30006178

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

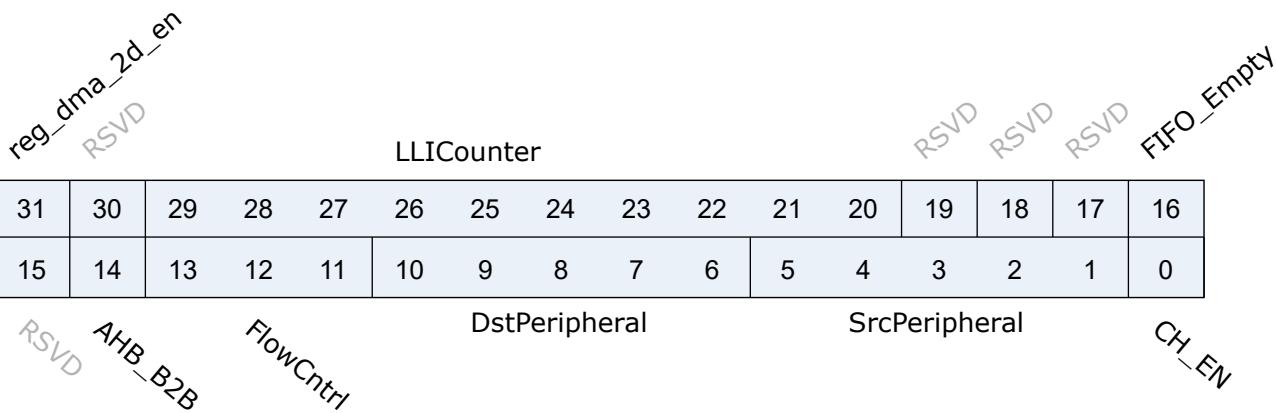
RSVD RSVD RSVD RSVD RSVD RSVD RSVD KEY\_STRB RSVD KEY\_MODE KEY\_EN

位	名称	权限	复位值	描述
31:8	RSVD			
7:4	KEY_STRB	r/w	4'h0	According Key mode set Key STRB Key mode : Key STRB 2'd0 : 4'h1 2'd1 : 4'h3 2'd2 : 4'h7 2'd3 : 4'hf
3	RSVD			

位	名称	权限	复位值	描述
2:1	KEY_MODE	r/w	0	Key Mode 2'd0 : 8-bit mode 2'd1 : 16-bit mode 2'd2 : 24-bit mode 2'd3 : 32-bit mode
0	KEY_EN	r/w	0	Key enable

#### 8.4.23 DMA2D\_C0\_CFG

地址: 0x3000617c



位	名称	权限	复位值	描述
31	reg_dma_2d_en	r/w	0	DMA 2d enable
30	RSVD			
29:20	LLICounter	r	0	LLI counter. Increased 1 each LLI run. Cleared 0 when config Control.
19:17	RSVD			
16	FIFO_Empty	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
15	RSVD			
14	AHB_B2B	r/w	0	AHB back to back enable

位	名称	权限	复位值	描述
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) The following only work when reg_dma_2d_en=0 : 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	CH_EN	r/w	0	Channel enable.

## 9.1 简介

LZ4 是一种无损数据压缩算法，其最大的特点是解压缩速度很快，但是压缩比一般。LZ4D 是一个针对使用 LZ4 算法压缩后的数据块进行解压缩的功能模块。其包含状态及控制寄存器组，用于设置输入压缩数据的起始地址，解压缩后的数据的输出地址，以及解压缩状态及可选的中断配置。

## 9.2 主要特征

- 支持标准 LZ4 压缩算法压缩的数据块
- 支持对不同压缩级别的压缩数据块的解压缩
- 支持多个 block 的压缩数据块的解压
- 解压缩完成、出错的各种事件标志
- 在事件发生时产生对应中断

## 9.3 功能描述

模块通过总线访问，从源地址获取压缩的数据块，通过内部算法进行解压缩之后，将解压缩的数据写入到指定的目标地址。在解压缩过程中，模块会更新相关的寄存器，并在某些事件发生时，设置对应的标志，例如解压缩成功、错误等。这些状态进而可以向 CPU 申请中断。

## 9.4 时钟

LZ4D 模块需要一路工作时钟。

## 9.5 编程流程

### 9.5.1 解压缩数据

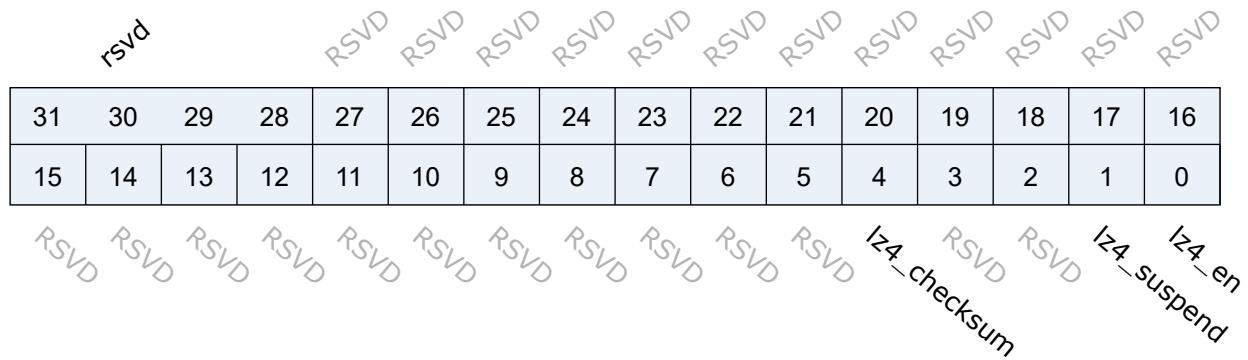
- 打开模块时钟
- 首先禁用模块，然后将需要解压缩的原始数据(不包含压缩文件头)地址写入到源数据起始地址寄存器中
- 准备足够大的连续内存，用于保存解压缩后的数据，长度建议对齐到4字节边界。将此内存块的地址写入到目的数据起始地址寄存器中
- 如果需要使用中断，那么需要安装中断处理函数，并设置对应的中断使能位
- 在控制寄存器中，使能模块，其就会开始解压缩
- 如果使能了完成中断，那么等待完成中断
- 如果没有使用中断，那么可以通过读取状态寄存器来判断的方式，等待解压缩完成

## 9.6 寄存器描述

名称	描述
lz4_config	
lz4_src_fix	
lz4_dst_fix	
lz4_src_start	
lz4_src_end	
lz4_dst_start	
lz4_dst_end	
lz4_int_en	
lz4_int_sta	
lz4_monitor	

### 9.6.1 lz4\_config

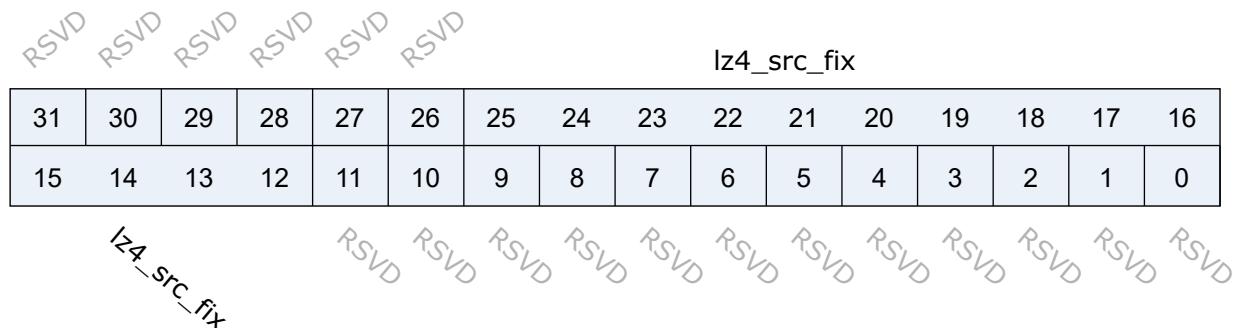
地址: 0x2000ad00



位	名称	权限	复位值	描述
31:28	rsvd	rsvd	0	
27:5	RSVD			
4	lz4_checksum	r/w	0	LZ4 block has checksum or not (info from frame header checksum flag)
3:2	RSVD			
1	lz4_suspend	r/w	0	LZ4 blocks decode suspend (FSM idle)
0	lz4_en	r/w	0	LZ4 blocks decode enable (each block has size/sequences/checksum)

### 9.6.2 lz4\_src\_fix

地址: 0x2000ad04



位	名称	权限	复位值	描述
31:26	RSVD			
25:12	lz4_src_fix	r/w	0	LZ4 source address fix bits

位	名称	权限	复位值	描述
11:0	RSVD			

### 9.6.3 lz4\_dst\_fix

地址: 0x2000ad08

lz4_dst_fix															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>lz4_dst_fix</i>															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

位	名称	权限	复位值	描述
31:26	RSVD			
25:12	lz4_dst_fix	r/w	0	LZ4 destination address fix bits
11:0	RSVD			

### 9.6.4 lz4\_src\_start

地址: 0x2000ad10

lz4_src_base								lz4_src_start							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>lz4_src_start</i>															

位	名称	权限	复位值	描述
31:26	lz4_src_base	r/w	0	LZ4 source address base (fix)
25:0	lz4_src_start	r/w	0	LZ4 source address start

## 9.6.5 lz4\_src\_end

地址: 0x2000ad14

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	lz4_src_end											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
lz4_src_end																	

位	名称	权限	复位值	描述
31:26	RSVD			
25:0	lz4_src_end	r	0	LZ4 source address end (Max. 64MB)

## 9.6.6 lz4\_dst\_start

地址: 0x2000ad18

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	lz4_dst_start											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
lz4_dst_start																	

位	名称	权限	复位值	描述
31:26	lz4_dst_base	r/w	0	LZ4 destination address base (fix)
25:0	lz4_dst_start	r/w	0	LZ4 destination address start

## 9.6.7 lz4\_dst\_end

地址: 0x2000ad1c

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	lz4_dst_end											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
lz4_dst_end																	

位	名称	权限	复位值	描述
31:26	RSVD			
25:0	lz4_dst_end	r	0	LZ4 destination address end (Max. 64MB)

### 9.6.8 lz4\_int\_en

地址: 0x2000ad20

lz4_dst_int_inv															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lz4_dst_int_en															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	lz4_err_en	lz4_done_en

位	名称	权限	复位值	描述
31:26	lz4_dst_int_inv	r/w	0	0: high int 1: low int
25:16	RSVD			
15:10	lz4_dst_int_en	r/w	0	
9:2	RSVD			
1	lz4_err_en	r/w	1	
0	lz4_done_en	r/w	1	

### 9.6.9 lz4\_int\_sta

地址: 0x2000ad24

lz4_dst_int_sta															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lz4_err_sta															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

位	名称	权限	复位值	描述
31:16	RSVD			
15:10	lz4_dst_int_sta	r	0	LZ4 destination address bit change
9:2	RSVD			
1	lz4_err_sta	r	0	
0	lz4_done_sta	r	0	LZ4 done status

## 9.6.10 lz4\_monitor

地址: 0x2000ad28

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

位	名称	权限	复位值	描述
31:4	RSVD			
3:0	lz4_cs	r	0	

## 10.1 简介

DBI 是显示总线接口 (Display Bus Interface) 的简称，也被称为 MCU 接口，是 MIPI 联盟定义的用于与显示屏通信的接口协议。DBI 协议的数据线和控制线是复用的，驱动的显示模块需要带有 GRAM。DBI 又可以细分为 MIPI DBI Type A、MIPI DBI Type B 和 MIPI DBI Type C 三种不同的模式，不同模式下的硬件接口以及数据采样都有所不同。如 MIPI DBI Type A 是下降沿采样 (基于 Motorola 6800 总线)，而 MIPI DBI Type B 是上升沿采样 (基于 Intel 8080 总线)。

## 10.2 主要特点

- 符合 MIPI® 联盟标准
- 支持 Type B 和 Type C 模式
- Type B 为 8 位数据接口
- Type C 支持 3-Line 和 4-Line
- 输入像素格式支持以下八种：
  - RGBA8888
  - BGRA8888
  - ARGB8888
  - ABGR8888
  - RGB888
  - BGR888
  - RGB565
  - BGR565
- 输出像素格式支持 RGB565、RGB666 和 RGB888
- 多种中断控制
- 支持 DMA 传输模式

## 10.3 功能描述

DBI 模块基本框图如图所示。

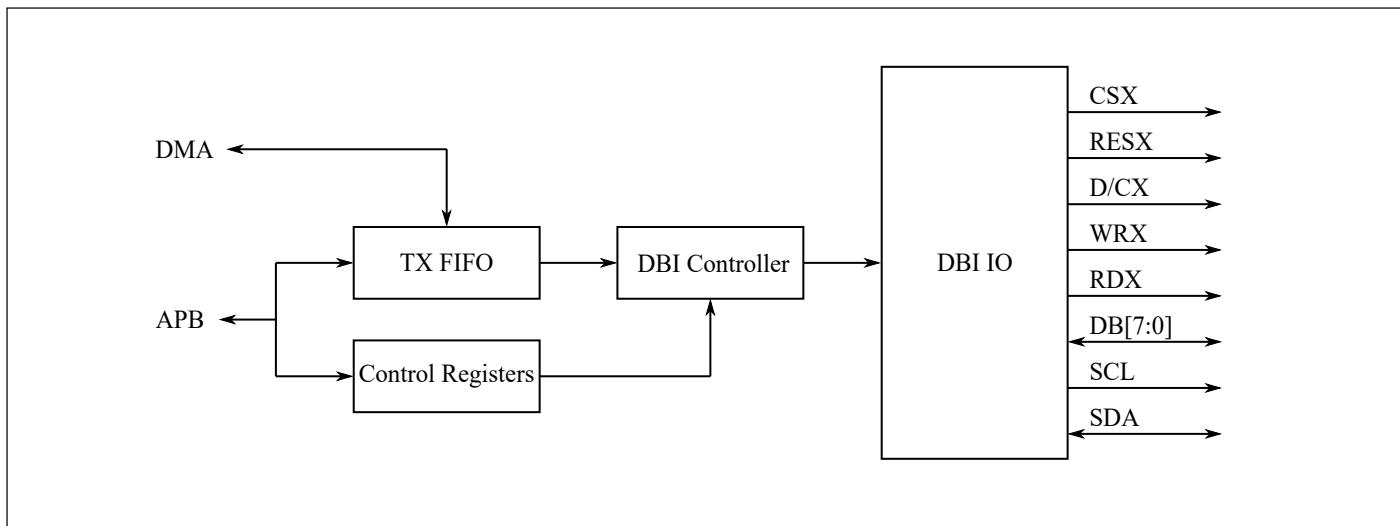


图 10.1: DBI 基本框图

### 10.3.1 DBI Type B

#### 10.3.1.1 写时序

DBI Type B 模式拥有 8 位并行数据线，MCU 向显示模块写入数据的时序如下图所示：

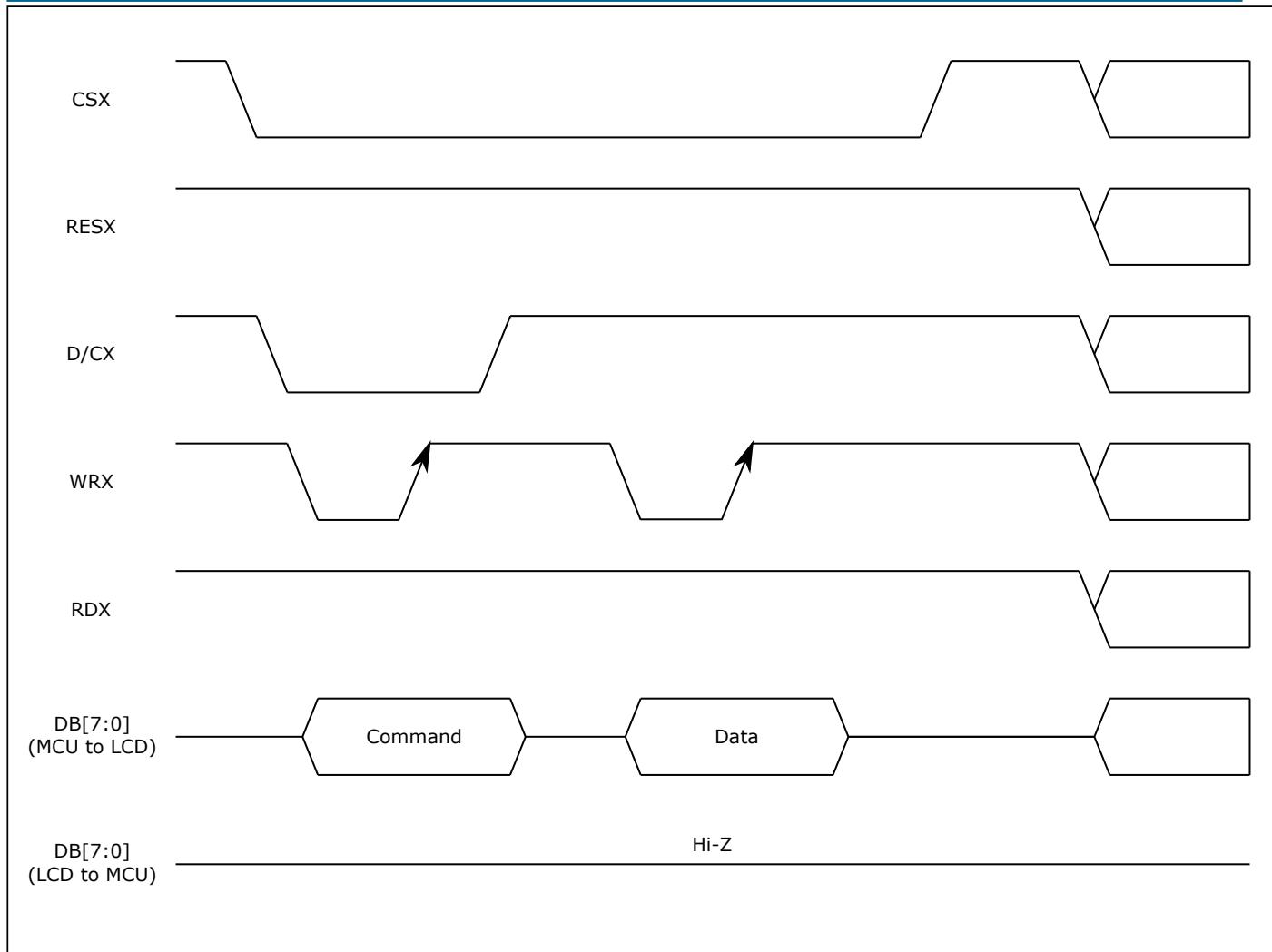


图 10.2: 写时序

其中：

- **CSX:** 片选信号。当该信号为低时显示模块被选中，为高时显示模块将忽略其他所有的接口信号；
- **RESX:** 外部复位信号。当该信号为低时显示模块被复位；
- **D/CX:** 数据/命令选择信号。当该信号为 0 时，DB[7:0] 位为命令；当该信号为 1 时，DB[7:0] 位为 RAM 数据或命令参数；
- **WRX:** 并行数据写入选通信号。该信号在写入周期内从高到低驱动，然后拉回到高。显示模块在该信号的上升沿读取 MCU 传输的信息。该信号是一种非同步信号，可在不使用时终止；
- **RDX:** 并行数据读取选通信号。该信号从高到低被驱动，然后在读取周期中被拉回到高。MCU 在该信号的上升沿读取显示模块传输的信息。该信号是一种非同步信号，可在不使用时终止；
- **DB[7:0]:** 8 位数据信号。用于传输命令、命令参数或数据。

### 10.3.1.2 读时序

MCU 从显示模块读取数据的时序如下图所示：

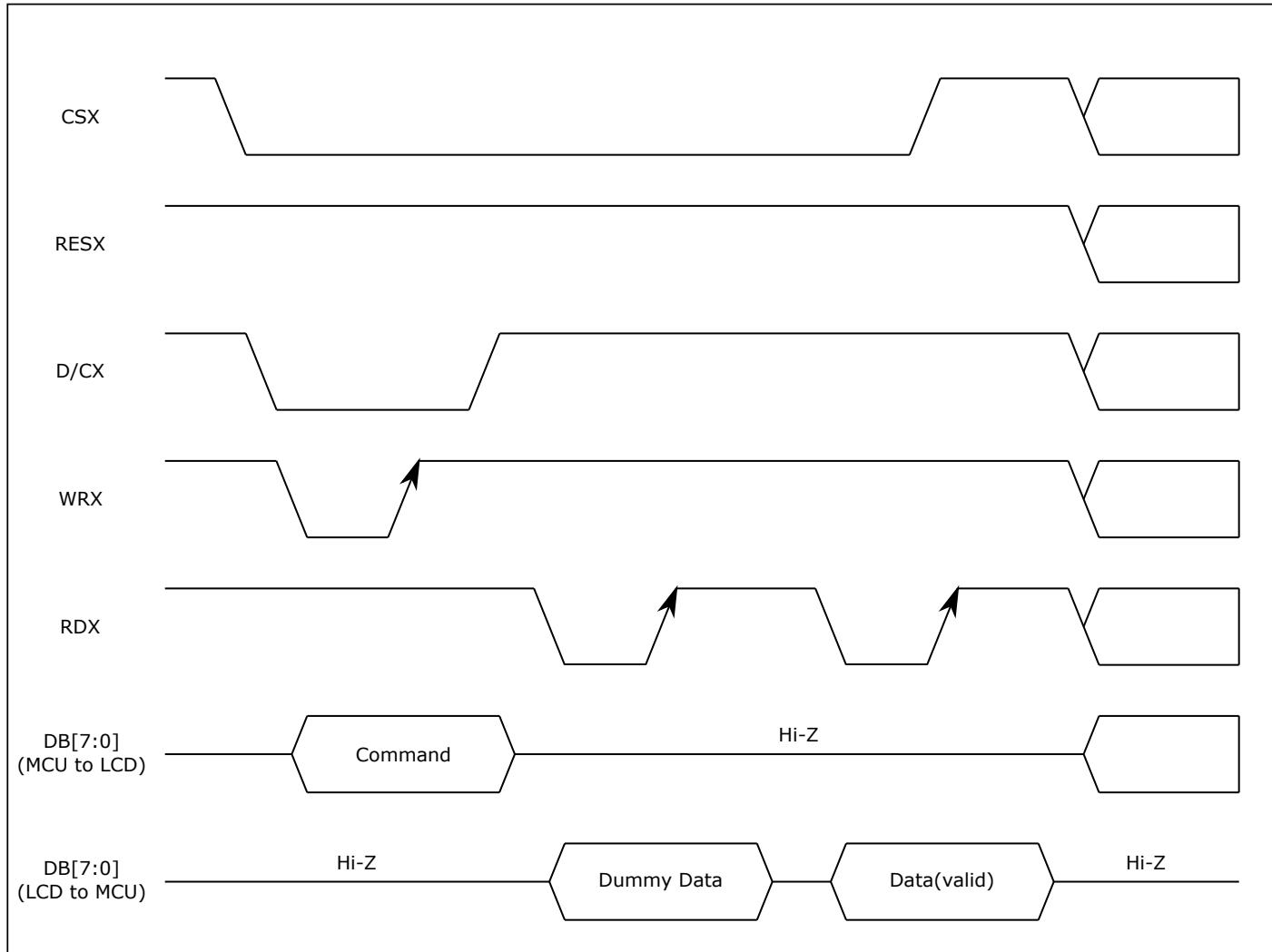


图 10.3: 读时序

### 10.3.1.3 输出 RGB565

RGB565 格式数据输出如下图所示：

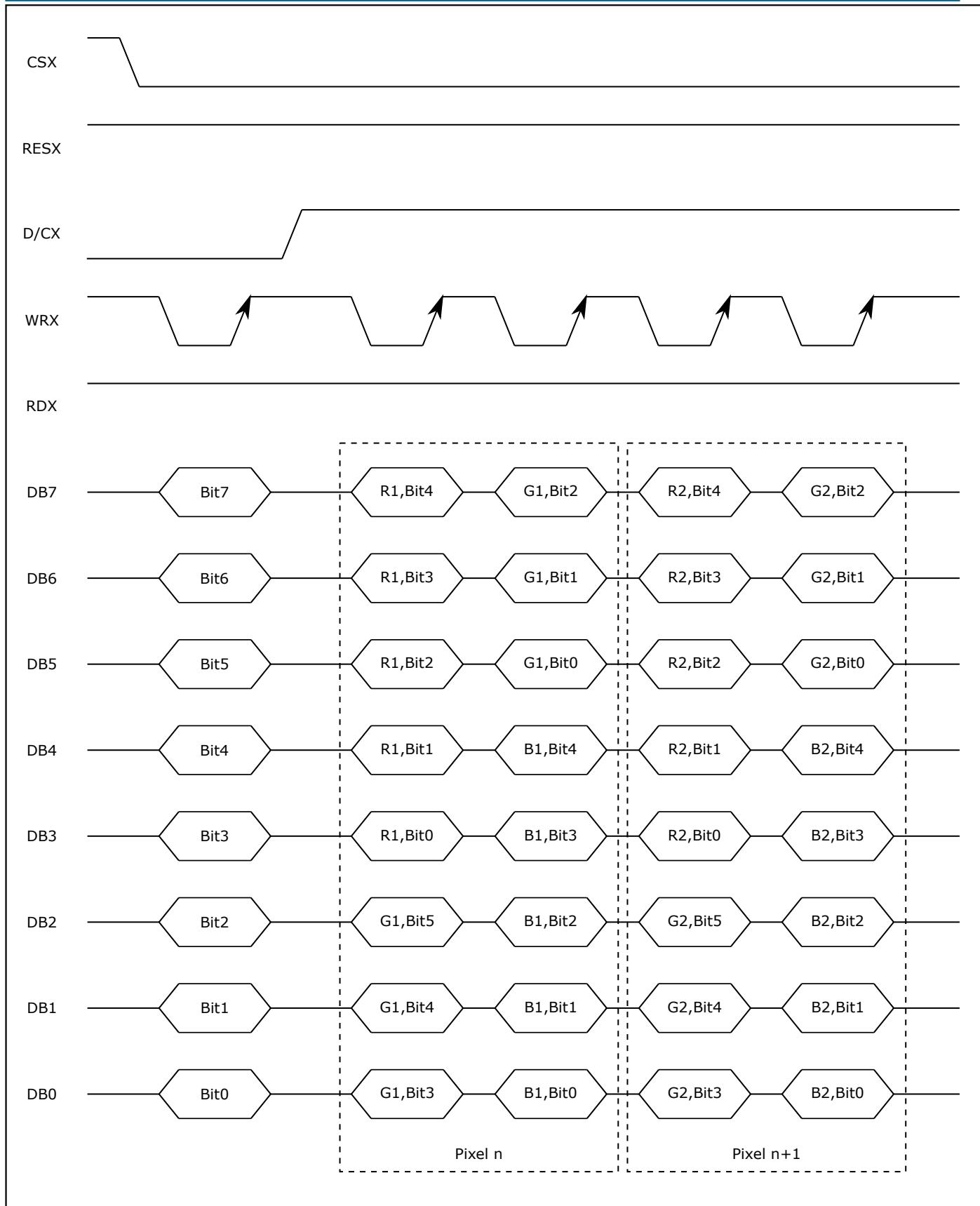


图 10.4: RGB565 输出

## 10.3.1.4 输出 RGB666

RGB666 格式数据输出如下图所示:

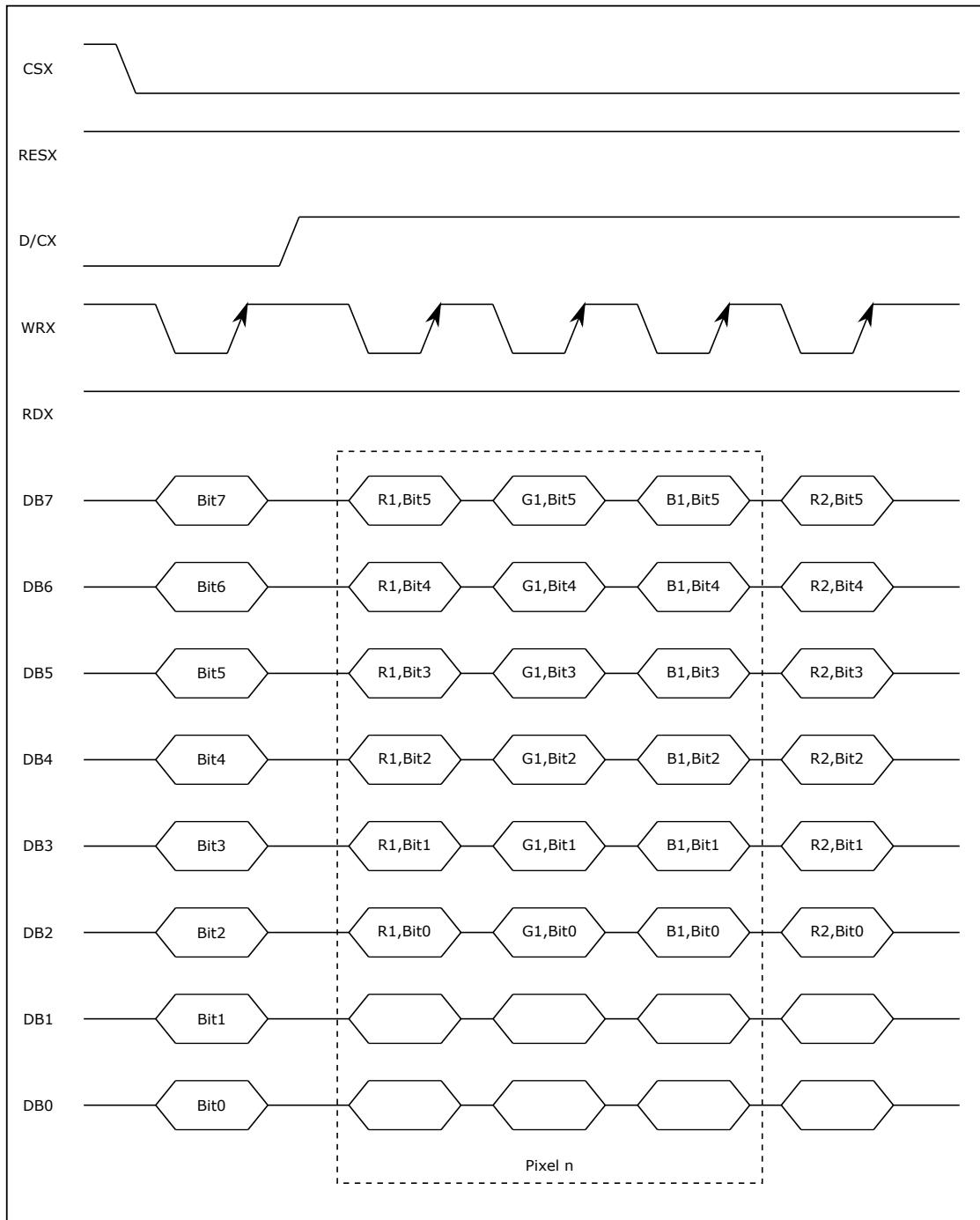


图 10.5: RGB666 输出

### 10.3.1.5 输出 RGB888

RGB888 格式数据输出如下图所示:

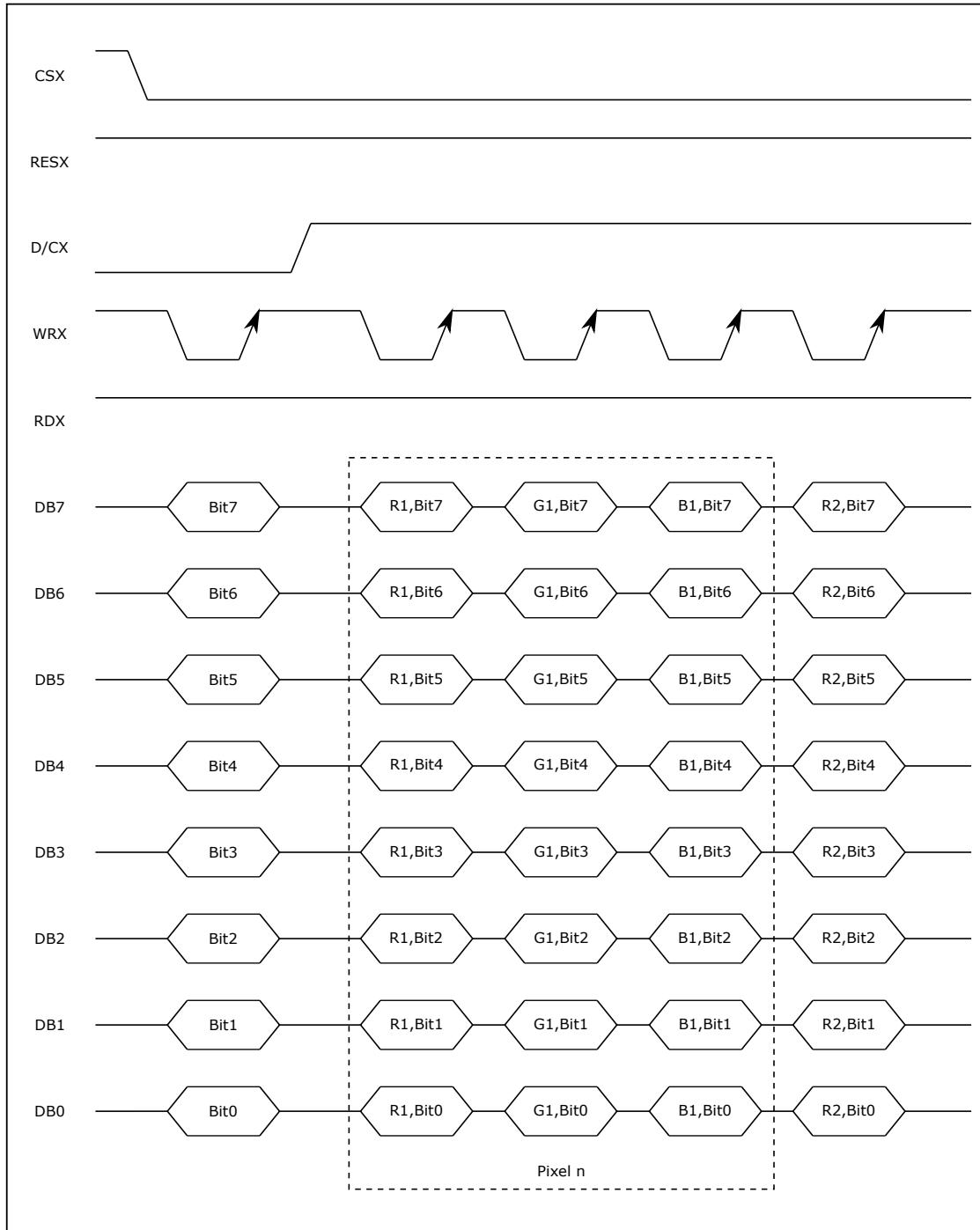


图 10.6: RGB888 输出

## 10.3.2 DBI Type C 3-Line

### 10.3.2.1 写时序

DBI Type C 模式是 3 线 9 位串行接口，MCU 向显示模块写入数据的时序如下图所示：

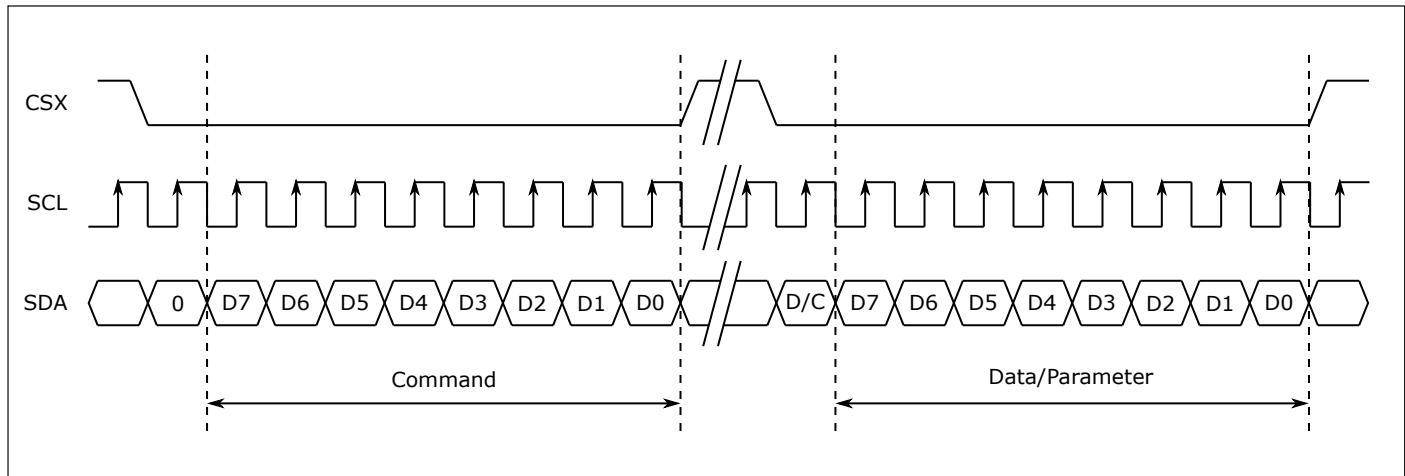


图 10.7: 写时序

其中：

- CSX: 片选信号。当该信号为低时显示模块被选中，为高时显示模块将忽略其他所有的接口信号；
- SCL: 串行时钟输入信号。用于为数据传输提供时钟信号。
- SDA: 串行数据输入/输出信号。在写操作中，串行数据包包含一个 D/CX(数据/命令) 选择位和一个传输字节。如果 D/CX 位为低，则传输字节为命令；如果 D/CX 位为高，则传输字节为显示数据或命令参数。

### 10.3.2.2 读时序

MCU 从显示模块读取数据的时序如下图所示：

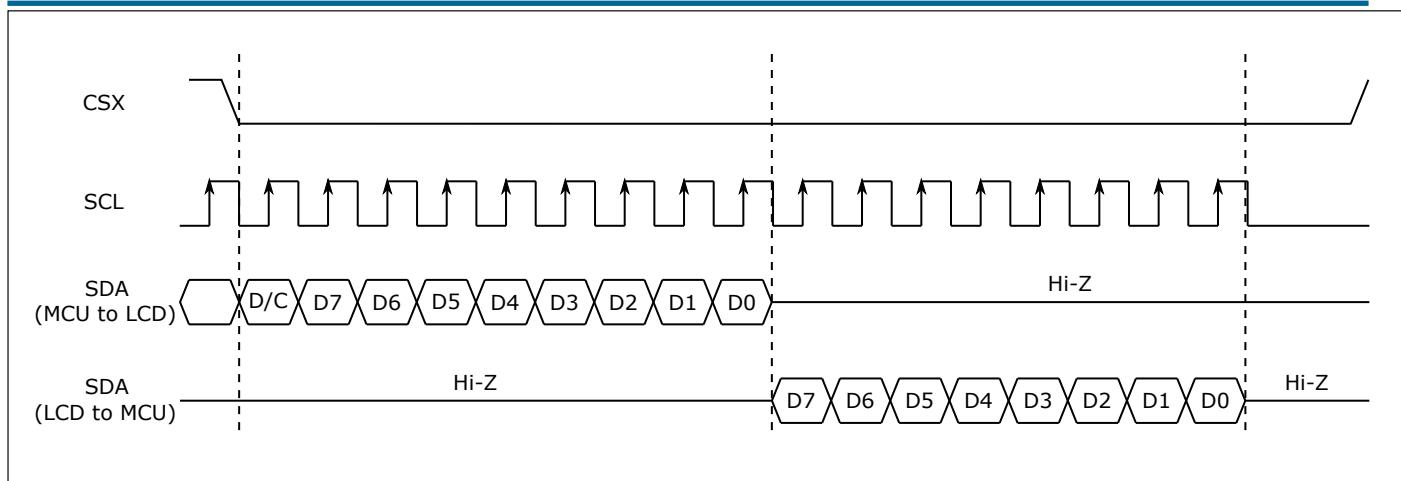


图 10.8: 读时序

### 10.3.2.3 输出 RGB565

RGB565 格式数据输出如下图所示:

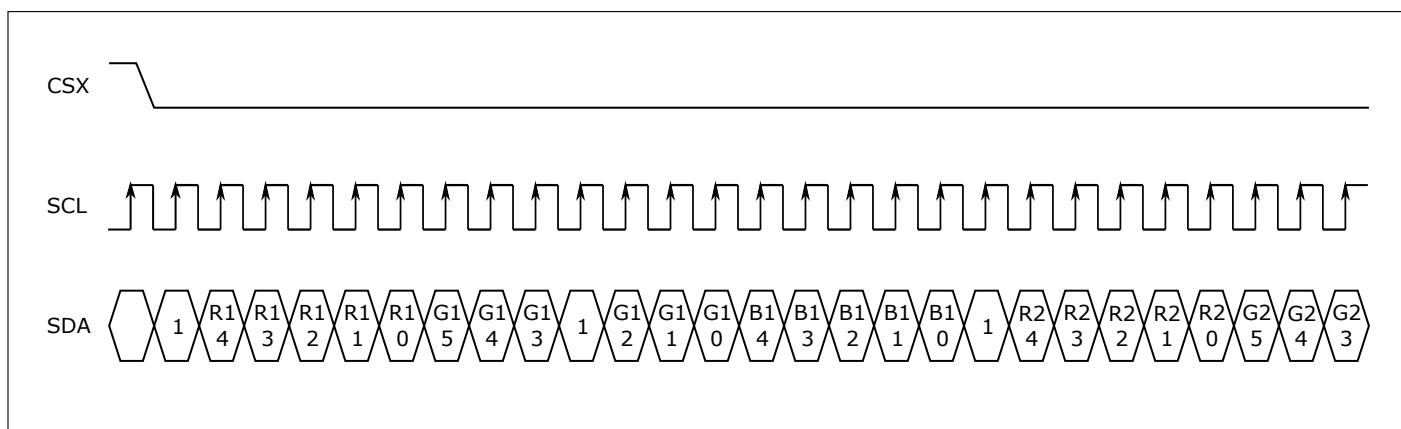


图 10.9: RGB565 输出

### 10.3.2.4 输出 RGB666

RGB666 格式数据输出如下图所示:

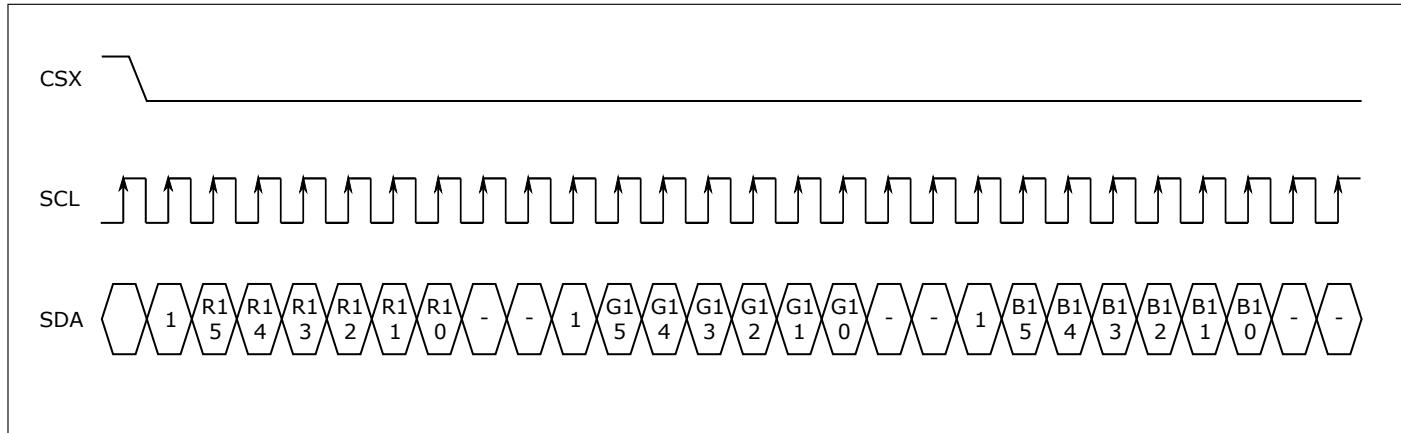


图 10.10: RGB666 输出

### 10.3.2.5 输出 RGB888

RGB888 格式数据输出如下图所示:

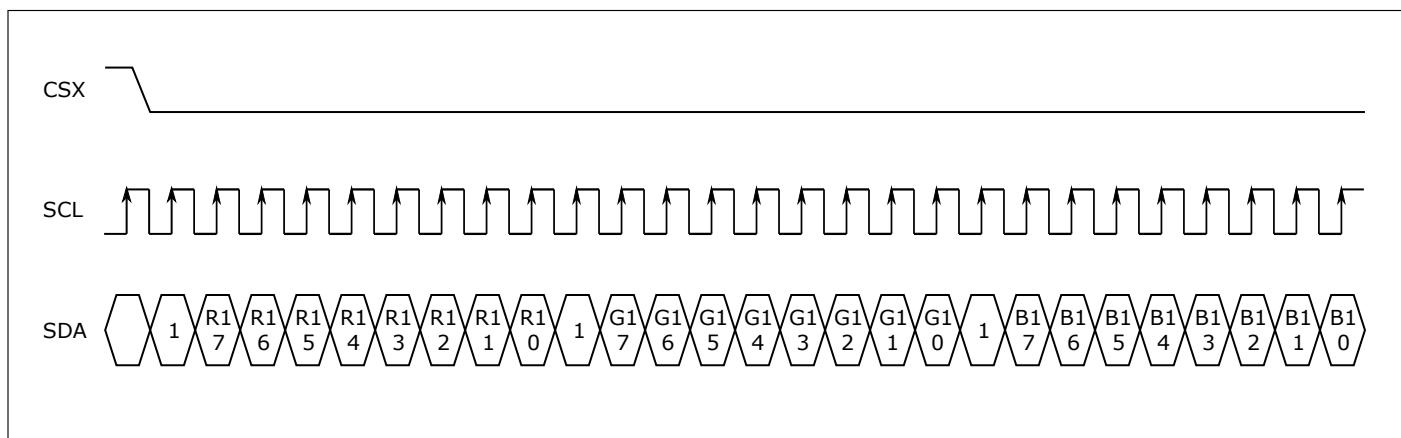


图 10.11: RGB888 输出

### 10.3.3 DBI Type C 4-Line

#### 10.3.3.1 写时序

DBI Type C 模式是 4 线 8 位串行接口，MCU 向显示模块写入数据的时序如下图所示:

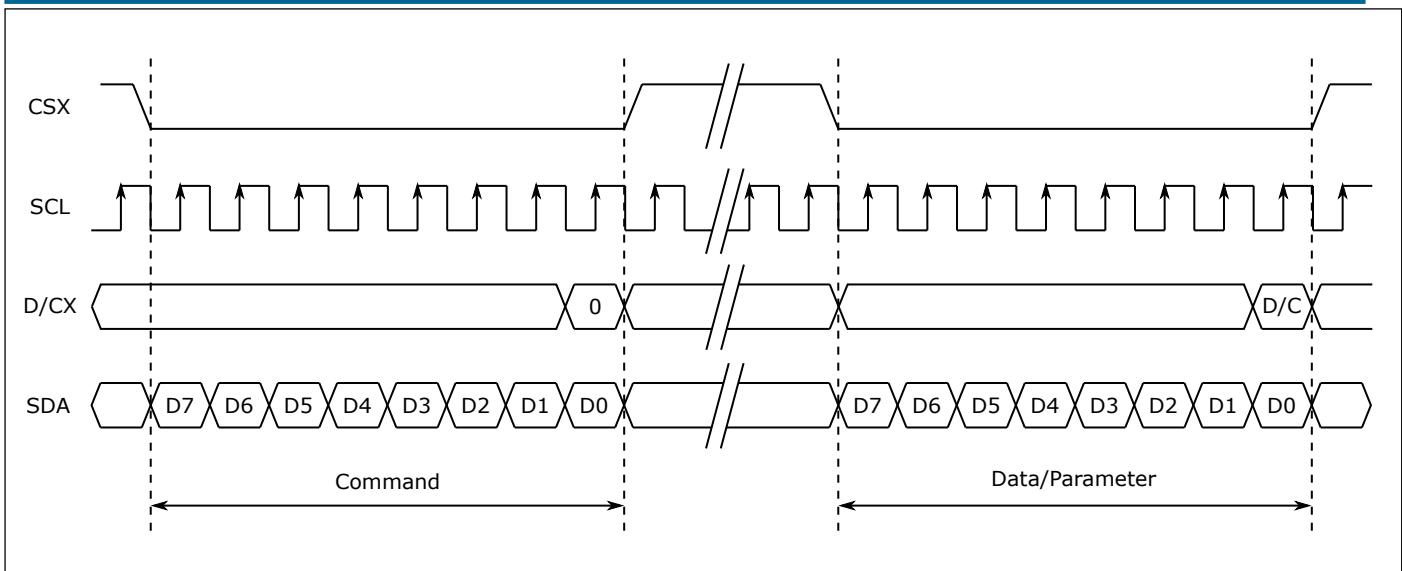


图 10.12: 写时序

其中：

- **CSX:** 片选信号。当该信号为低时显示模块被选中，为高时显示模块将忽略其他所有的接口信号；
- **D/CX:** 数据/命令选择信号。如果该信号为低，则 SDA 传输的信息为命令；如果该信号为高，则 SDA 传输的信息为显示数据或命令参数。
- **SCL:** 串行时钟输入信号。用于为数据传输提供时钟信号。
- **SDA:** 串行数据输入/输出信号。用于传输命令、命令参数或数据。

### 10.3.3.2 读时序

MCU 从显示模块读取数据的时序如下图所示：

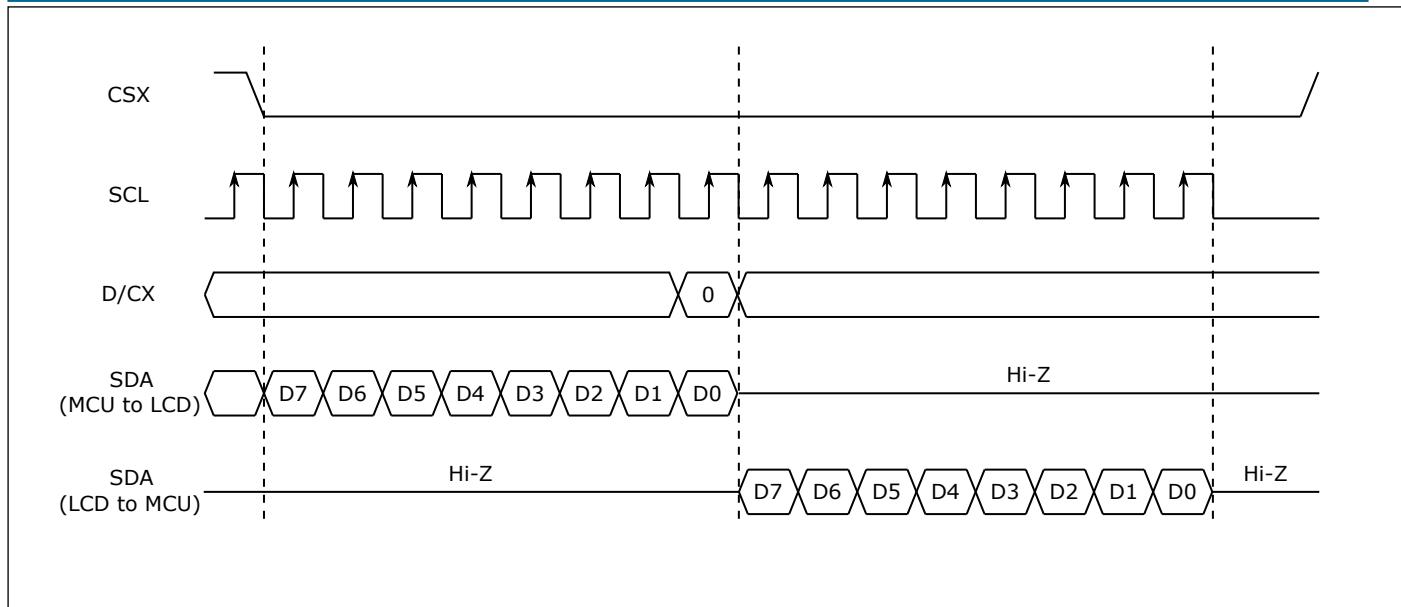


图 10.13: 读时序

### 10.3.3.3 输出 RGB565

RGB565 格式数据输出如下图所示：

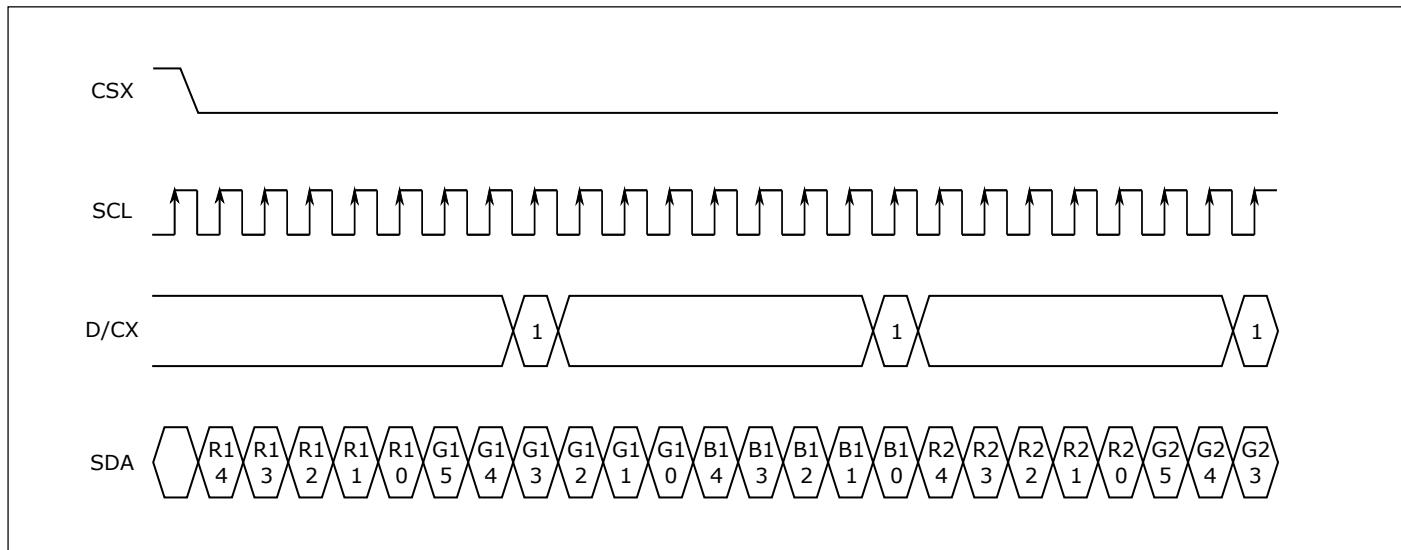


图 10.14: RGB565 输出

### 10.3.3.4 输出 RGB666

RGB666 格式数据输出如下图所示:

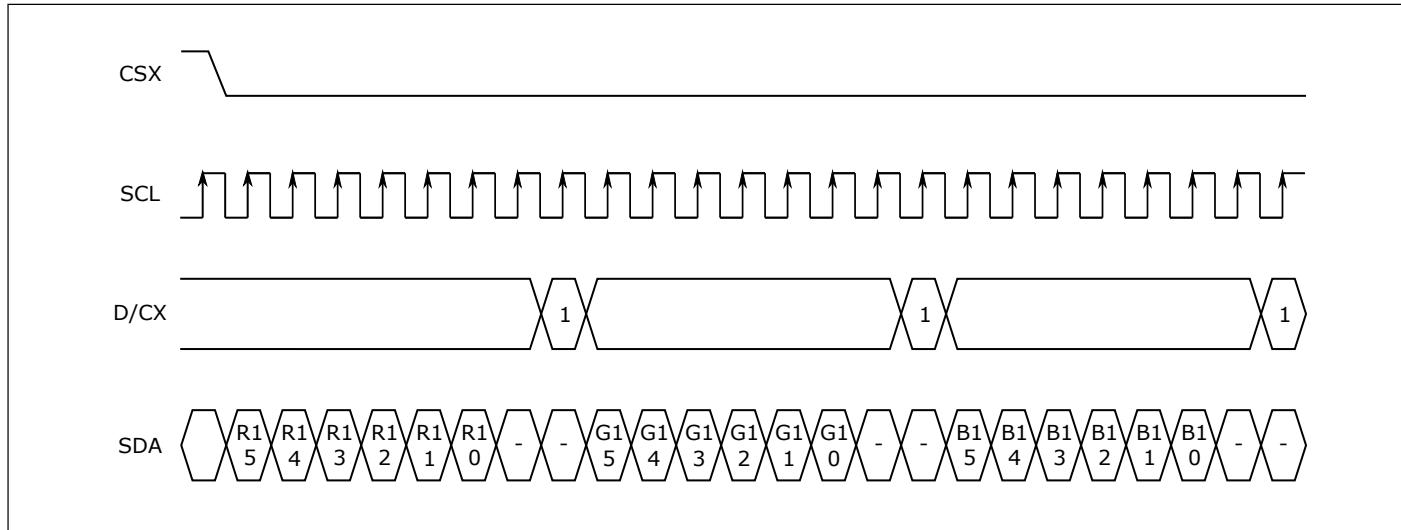


图 10.15: RGB666 输出

### 10.3.3.5 输出 RGB888

RGB888 格式数据输出如下图所示:

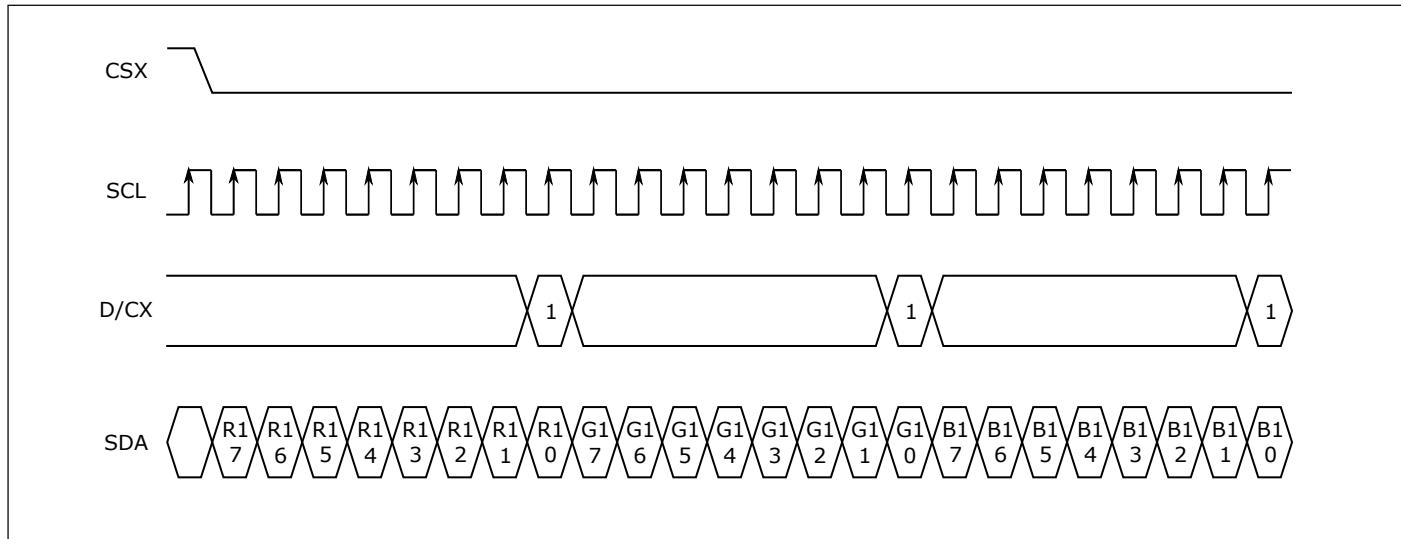


图 10.16: RGB888 输出

### 10.3.4 输入像素格式

DBI 模块支持 8 种不同的输入像素格式，各种格式在内存中的排布如下图所示：

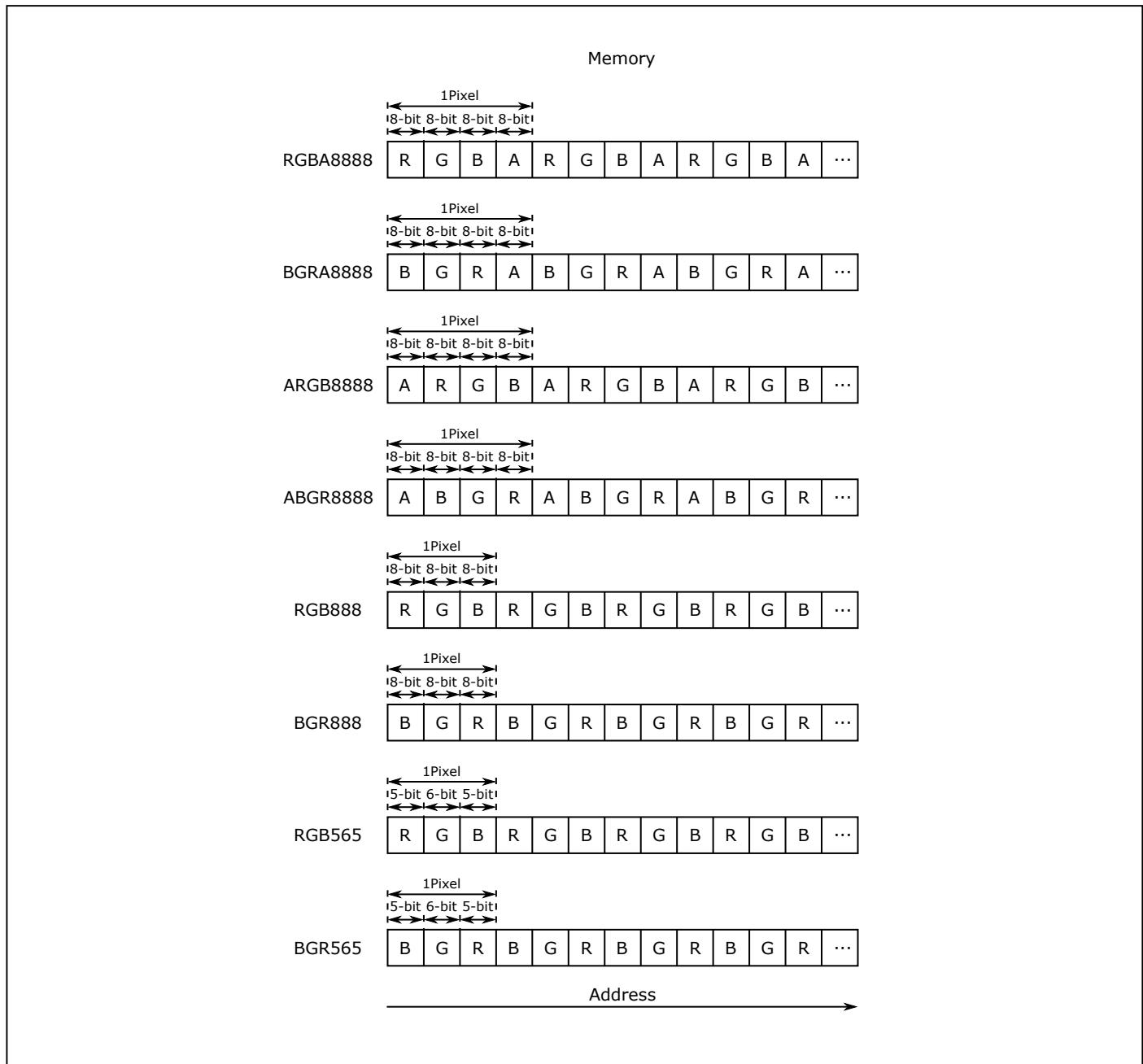


图 10.17: 输入像素格式

### 10.3.5 中断

DBI 具有以下几种中断：

- 传输结束中断
- TX FIFO 请求中断
- FIFO 溢出错误中断

通过设置一个像素计数值，传输结束中断会在传输的像素数据达到计数值时产生，Type B 和 Type C 都会产生该中断；

当 DBI\_FIFO\_CONFIG\_1 中 TFICNT 大于 TFITH 时，产生 TX FIFO 请求中断，当条件不满足时该中断标志会自动清除；

FIFO 溢出错误中断会在 TX 发生 Overflow 或者 Underflow 时产生。

### 10.3.6 DMA

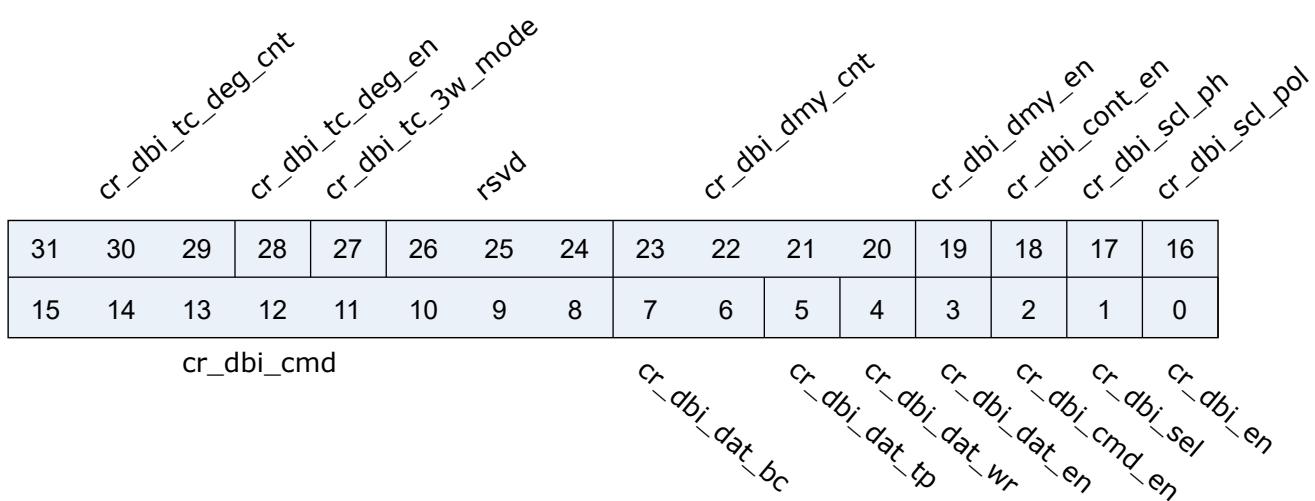
DBI TX 支持 DMA 传输模式，使用该模式需要通过寄存器 DBI\_FIFO\_CONFIG\_1 的位 <TFITH> 设置 TX FIFO 的阈值。当该模式启用后，如果 TFICNT 大于 TFITH，则会触发 DMA TX 请求，配置好 DMA 后，DMA 在收到该请求时，会按照设定从内存中将数据搬运到 TX FIFO。

## 10.4 寄存器描述

名称	描述
dbi_config	
dbi_int_sts	
dbi_bus_busy	
dbi_pix_cnt	
dbi_prd	
dbi_wdata	
dbi_rdata	
dbi_fifo_config_0	
dbi_fifo_config_1	
dbi_fifo_wdata	

### 10.4.1 dbi\_config

地址: 0x3001b000

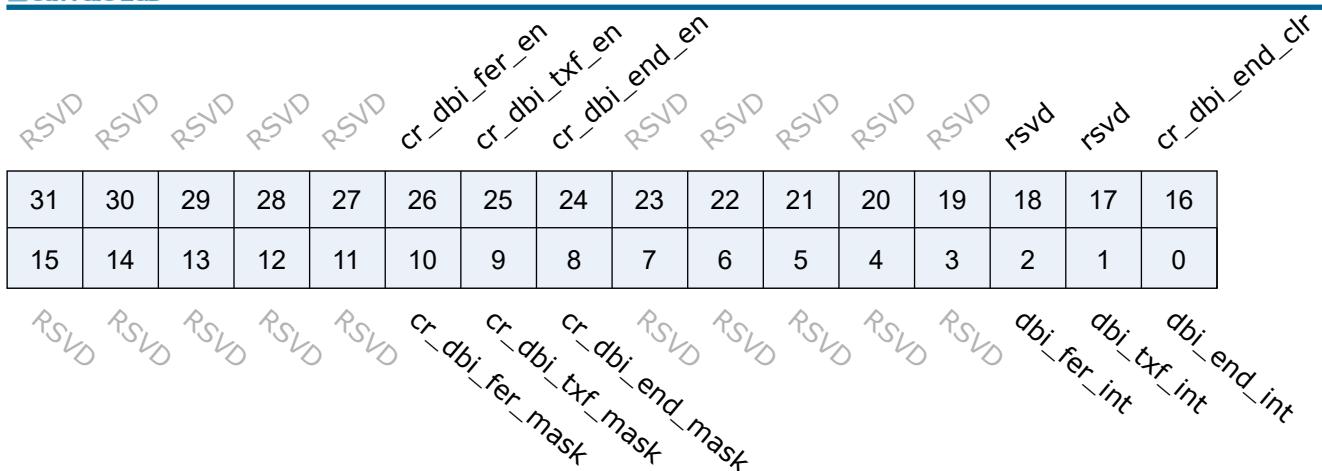


位	名称	权限	复位值	描述
31:29	cr_dbi_tc_deg_cnt	r/w	3'd0	De-glitch function cycle count
28	cr_dbi_tc_deg_en	r/w	1'b0	Enable signal of all input de-glitch function
27	cr_dbi_tc_3w_mode	r/w	1'b0	DBI Type C 3-wire mode enable 1'b0: 4-wire mode is enabled 1'b1: 3-wire mode is enabled
26:24	rsrd	rsrd	3'h0	
23:20	cr_dbi_dmy_cnt	r/w	4'd0	Dummy cycle count, unit: period defined by dbi_prd_d Effective only in Type C (Fixed to 1 dbi_prd_d period in Type B)
19	cr_dbi_dmy_en	r/w	1'b0	Enable signal of dummy cycle(s) 1'b0: Disabled, no dummy cycle(s) between command phase and data phase 1'b1: Enabled, dummy cycle(s) will be inserted between command phase and data phase
18	cr_dbi_cont_en	r/w	1'b1	Enable signal of pixel data continuous transfer mode 1'b0: Disabled, CS_n will de-assert between each pixel 1'b1: Enabled, CS_n will stay asserted between each consecutive pixel
17	cr_dbi_scl_ph	r/w	1'b0	SCL clock phase inverse signal

位	名称	权限	复位值	描述
16	cr_dbi_scl_pol	r/w	1'b1	SCL clock polarity 0: SCL output LOW at IDLE state 1: SCL output HIGH at IDLE state
15:8	cr_dbi_cmd	r/w	8'h2C	DBI Command
7:6	cr_dbi_dat_bc	r/w	2'd0	Data byte count of normal data (pixel data count is determined by cr_dbi_pix_cnt)
5	cr_dbi_dat_tp	r/w	1'b0	Data type select 1'b0: Normal data (parameter) 1'b1: Pixel data Note: Read command supports normal data only
4	cr_dbi_dat_wr	r/w	1'b1	Data phase Read/Write select 1'b0: Read data 1'b1: Write data
3	cr_dbi_dat_en	r/w	1'b1	Data enable signal 1'b0: Data phase disabled 1'b1: Data phase enabled
2	cr_dbi_cmd_en	r/w	1'b1	Command enable signal 1'b0: No command phase 1'b1: Command will be sent
1	cr_dbi_sel	r/w	1'b0	Select signal of DBI Type B or C 1'b0: DBI Type B 1'b1: DBI Type C
0	cr_dbi_en	r/w	1'b0	Enable signal of DBI function Asserting this bit will trigger the transaction, and should be de-asserted after finish

#### 10.4.2 dbi\_int\_sts

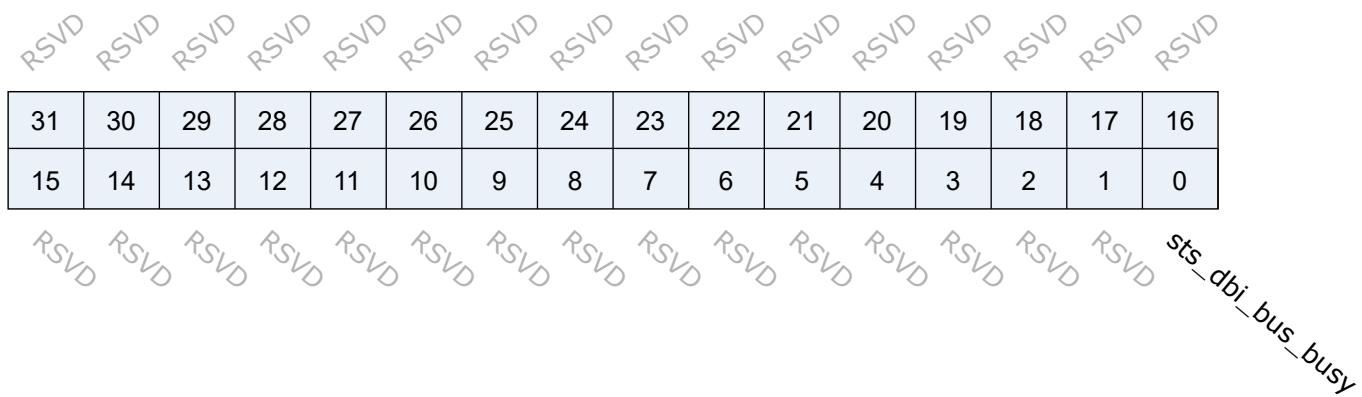
地址: 0x3001b004



位	名称	权限	复位值	描述
31:27	RSVD			
26	cr_db_i_fer_en	r/w	1'b1	Interrupt enable of dbi_fer_int
25	cr_db_i_txf_en	r/w	1'b1	Interrupt enable of dbi_txe_int
24	cr_db_i_end_en	r/w	1'b1	Interrupt enable of dbi_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_db_i_end_clr	w1c	1'b0	Interrupt clear of dbi_end_int
15:11	RSVD			
10	cr_db_i_fer_mask	r/w	1'b1	Interrupt mask of dbi_fer_int
9	cr_db_i_txf_mask	r/w	1'b1	Interrupt mask of dbi_txe_int
8	cr_db_i_end_mask	r/w	1'b1	Interrupt mask of dbi_end_int
7:3	RSVD			
2	dbi_fer_int	r	1'b0	TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	dbi_txf_int	r	1'b1	TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	dbi_end_int	r	1'b0	Transfer end interrupt, shared by both Type B and C mode

### 10.4.3 dbi\_bus\_busy

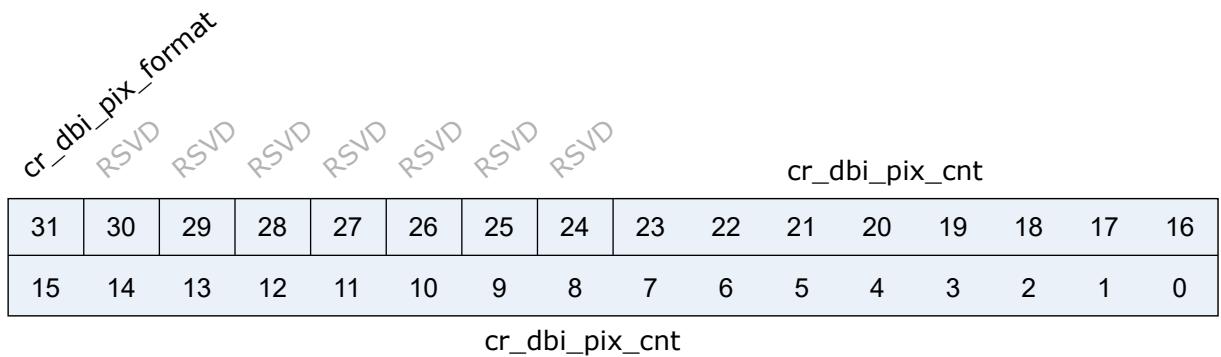
地址: 0x3001b008



位	名称	权限	复位值	描述
31:1	RSVD			
0	sts_dbi_bus_busy	r	1'b0	Indicator of dbi bus busy

### 10.4.4 dbi\_pix\_cnt

地址: 0x3001b00c



位	名称	权限	复位值	描述
31	cr_dbi_pix_format	r/w	1'b0	Pixel format 1'b0: RGB565 1'b1: RGB888/RGB666
30:24	RSVD			
23:0	cr_dbi_pix_cnt	r/w	24'h0	Pixel count

### 10.4.5 dbi\_prd

地址: 0x3001b010

cr_dbi_prd_d_ph_1								cr_dbi_prd_d_ph_0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_dbi_prd_i								cr_dbi_prd_s							

位	名称	权限	复位值	描述
31:24	cr_dbi_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 (please refer to "Timing" tab)
23:16	cr_dbi_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (please refer to "Timing" tab)
15:8	cr_dbi_prd_i	r/w	8'd15	Length of INTERVAL between pixel data (please refer to "Timing" tab)
7:0	cr_dbi_prd_s	r/w	8'd15	Length of START/STOP condition (please refer to "Timing" tab)

### 10.4.6 dbi\_wdata

地址: 0x3001b018

cr_dbi_wdata															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_dbi_wdata															

位	名称	权限	复位值	描述
31:0	cr_dbi_wdata	r/w	32'h0	Data to be written into display IC using write command

### 10.4.7 dbi\_rdata

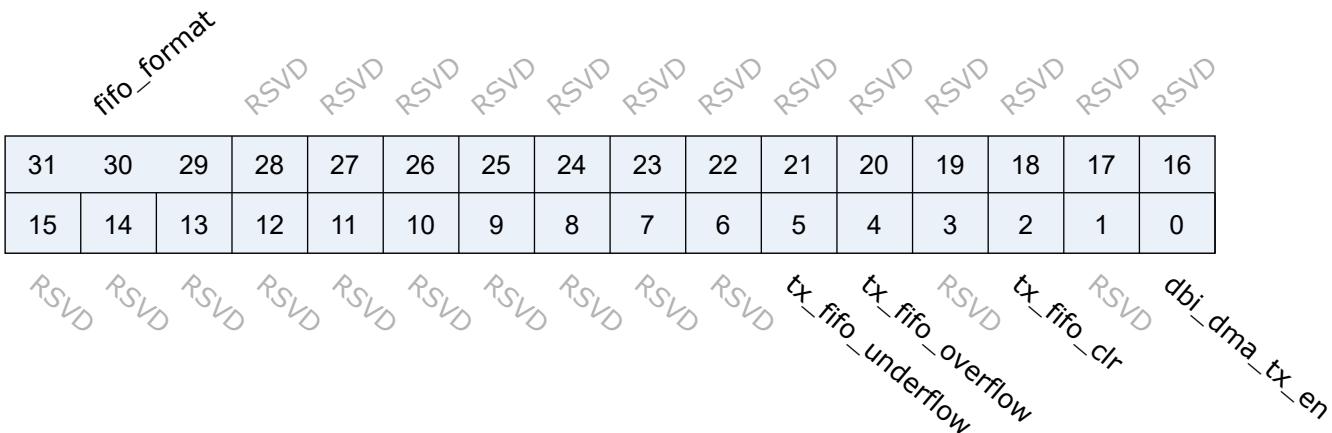
地址: 0x3001b01c

sts_dbi_rdata															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sts_dbi_rdata															

位	名称	权限	复位值	描述
31:0	sts_dbi_rdata	r	32'h0	Data read from display IC using read command

#### 10.4.8 dbi\_fifo\_config\_0

地址: 0x3001b080



位	名称	权限	复位值	描述
31:29	fifo_format	r/w	3'd0	FIFO data format (see Tab 'FIFO Format' for details) 3'd0: 8'h0, B[7:0], G[7:0], R[7:0] 3'd1: 8'h0, R[7:0], G[7:0], B[7:0] 3'd2: B[7:0], G[7:0], R[7:0], 8'h0 3'd3: R[7:0], G[7:0], B[7:0], 8'h0 3'd4: R[7:0], B[7:0], G[7:0], R[7:0] 3'd5: B[7:0], R[7:0], G[7:0], B[7:0] 3'd6: 2B[7:3], G[7:2], R[7:3] 3'd7: 2R[7:3], G[7:2], B[7:3]
28:6	RSVD			
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	RSVD			
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	RSVD			
0	dbi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

## 10.4.9 dbi\_fifo\_config\_1

地址: 0x3001b084

RSVD	tx_fifo_th														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:19	RSVD			
18:16	tx_fifo_th	r/w	3'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:4	RSVD			
3:0	tx_fifo_cnt	r	4'd8	TX FIFO available count

#### 10.4.10 dbi\_fifo\_wdata

地址: 0x3001b088

dbi\_fifo\_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dbi\_fifo\_wdata

位	名称	权限	复位值	描述
31:0	dbi_fifo_wdata	w	x	Pixel data with 8 types of format (determined by fifo_format)

# 11

DPI

## 11.1 简介

DPI 是显示像素接口 (Display Pixel Interface) 的简称，也被称为 RGB 接口，是 MIPI 联盟定义的用于与显示屏通信的接口协议。本模块拥有 16 位并行数据线，由系统内存充当显存，通过持续输出数据和同步信号，直接驱动显示屏进行显示。

## 11.2 主要特点

- 符合 MIPI® 联盟标准
- 支持单缓冲区和乒乓缓冲区作为显存
- 乒乓操作支持硬件控制和软件控制
- 支持测试模式
- 可编程定时参数
- 输入像素格式 (缓冲区数据格式) 支持：
  - YUV422(交织)
  - RGB888
  - RGB565
  - RGBA8888
  - YUV420(平面)
- 输出像素格式支持 RGB565
- 一个可编程中断
- 可与 OSD 配合使用

## 11.3 功能描述

DPI 模块基本框图如图所示。

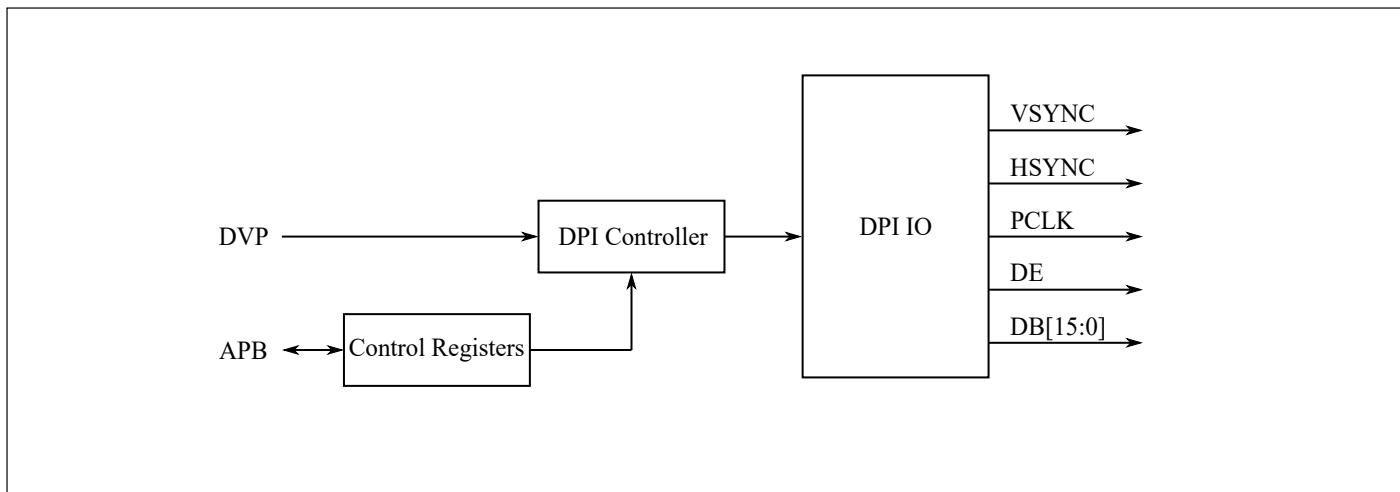


图 11.1: DPI 基本框图

### 11.3.1 DPI 时序

16 位 DPI 接口时序如下图所示：

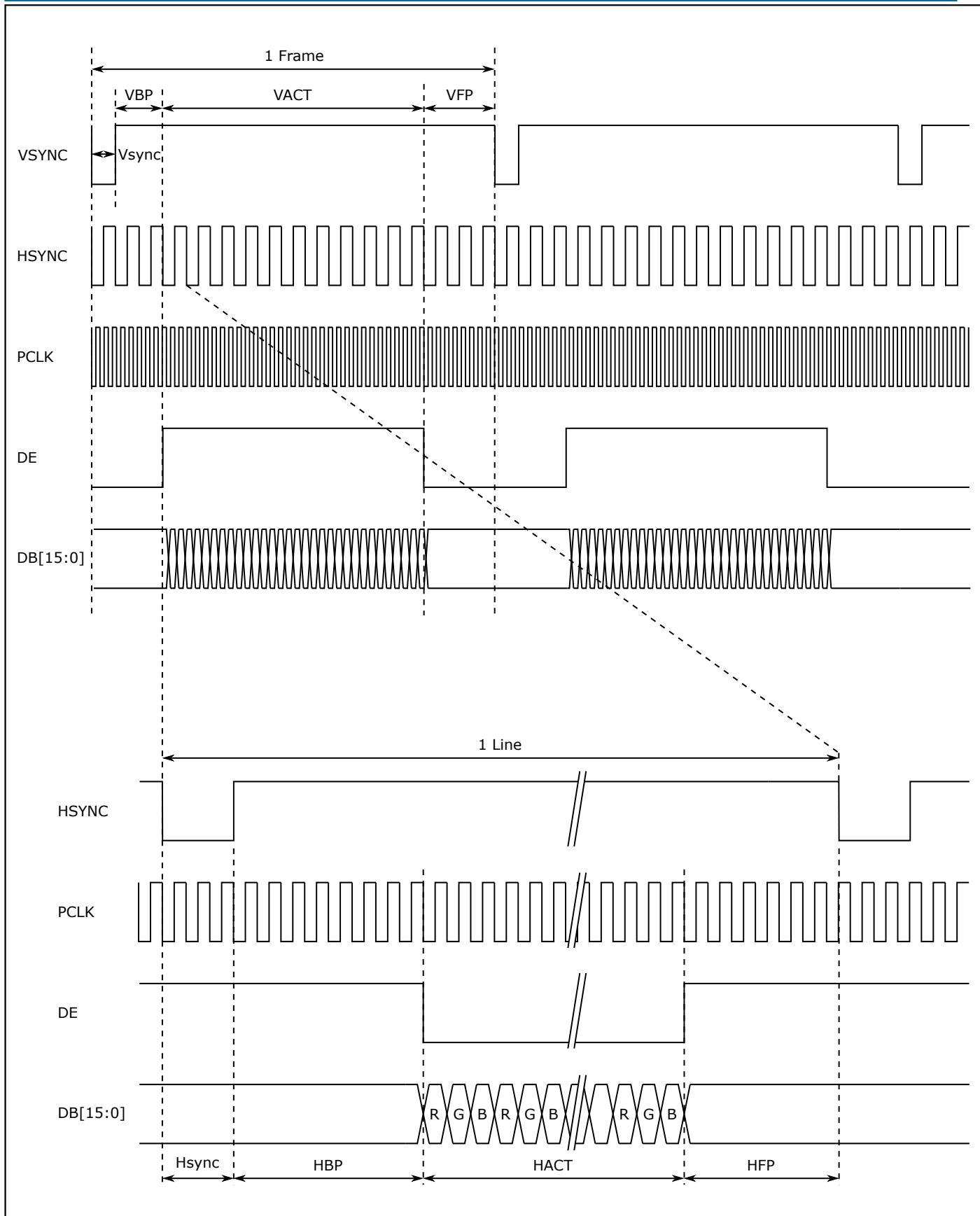


图 11.2: DPI 时序图

其中：

- VSYNC(Vertical Sync): 垂直同步定时信号，用于表示每帧显示图像的开始；
- HSYNC(Horizontal Sync): 水平同步定时信号，用于表示每行水平像素的开始；
- PCLK(Pixel Clock): 像素时钟，持续不断地产生，VSYNC、HSYNC、DE 和 DB[15:0] 会在 PCLK 的上升沿被采样；
- DE(Data Enable): 数据使能信号，用于指示有效像素数据；
- DB[15:0](Data Bit): 并行数据位，总共有 16 位，用于传输像素数据。

### 11.3.2 可编程定时参数

DPI 的定时参数如下图所示：

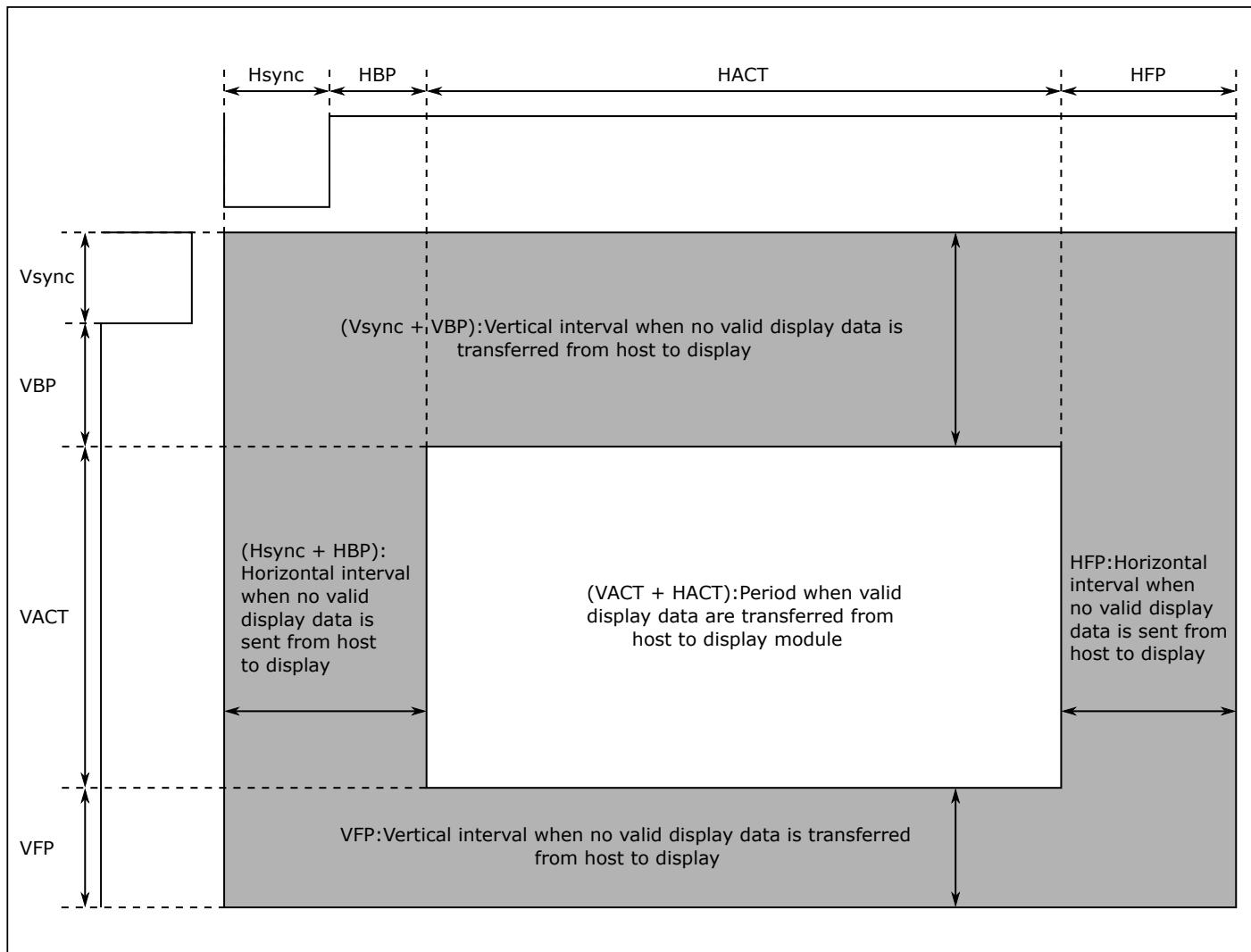


图 11.3: 定时参数图

垂直周期 (一帧) 等于 Vsync、VBP、VACT 与 VFP 之和；水平周期 (一行) 等于 Hsync、HBP、HACT 与 HFP 之和。定时参数的可编程范围如下表所示：

Parameters	Symbols	Min.	Step	Max.	Unit
Horizontal Synchronization	Hsync	1	1	255	PCLK Cycle
Horizontal Back Porch	HBP	1	1	-	PCLK Cycle
Horizontal Active	HACT	176	1	800	PCLK Cycle
Horizontal Front Porch	HFP	1	1	255	PCLK Cycle
Vertical Synchronization	Vsync	1	1	255	Line
Vertical Back Porch	VBP	1	1	-	Line
Vertical Active	VACT	208	1	480	Line
Vertical Front Porch	VFP	1	1	255	Line

图 11.4: 定时参数取值范围表

### 11.3.3 单缓冲区

DPI 模块可以使用一块大小为显示分辨率 \* 每个像素所占字节数的内存作为显存，当开始工作时，DPI 会重复不断地将这块内存中的数据传输给显示模块。如果对这块内存中的数据进行修改，则修改内容可能在当前帧生效，也可能在下一帧生效，这取决于修改的数据在内存中的位置位于 DPI 当前传输的数据在内存中的位置之后还是之前。

### 11.3.4 乒乓缓冲区

DPI 模块可以使用两块大小分别为显示分辨率 \* 每个像素所占字节数的内存作为显存，并重复不断地将其中一块内存中的数据传输给显示模块。将当前传输的内存从一块切换到另一块的控制方式可以设置为硬件控制或软件控制。当设置为硬件控制时，DPI 的数据由 Cam 模块提供（参阅 Cam 模块介绍），DPI 会在 Cam 模块将一帧数据完整写入没有参与传输的那块内存，并且当前正在传输的内存块中的数据传输完成之后将使用的内存块切换为另一块。当设置为软件控制时，DPI 当前用于传输的内存块由 `<SWAP_ID_SWV>` 进行设置，写 0 和写 1 分别对应两个内存块。

### 11.3.5 测试模式

测试模式下，DPI 可以输出同一行像素颜色相同、不同行之间颜色渐变的图像。测试模式需要设置 3 个参数：像素数据最小值 `VAL_MIN(16-bit)`、像素数据最大值 `VAL_MAX(16-bit)` 和步长 `STEP(8-bit)`。DPI 会以 `VAL_MIN` 作为第 1 行每个像素的值，`(VAL_MIN+STEP)` 作为第 2 行每个像素的值，`(VAL_MIN+STEP*2)` 作为第 3 行每个像素的值... 以此类推，直到第 N 行 `(VAL_MIN+STEP*(N-1))` 大于 `VAL_MAX` 时回到 `VAL_MIN`，循环往复直到输出的行数达到设置值。以 `VAL_MIN=0x00E0`, `VAL_MAX=0x012F`, `STEP=0x10` 为例，测试模式下显示屏的显示如下图所示：

VAL_MIN=0x00E0, VAL_MAX=0x012F, STEP=0x10	
0x 00E0 00E0 00E0 00E0 00E0 ... 00E0 00E0	Line 1
0x 00F0 00F0 00F0 00F0 00F0 ... 00F0 00F0	Line 2
0x 0100 0100 0100 0100 0100 ... 0100 0100	Line 3
0x 0110 0110 0110 0110 0110 ... 0110 0110	Line 4
0x 0120 0120 0120 0120 0120 ... 0120 0120	Line 5
0x 00E0 00E0 00E0 00E0 00E0 ... 00E0 00E0	Line 6
0x 00F0 00F0 00F0 00F0 00F0 ... 00F0 00F0	Line 7
0x 0100 0100 0100 0100 0100 ... 0100 0100	Line 8
0x 0110 0110 0110 0110 0110 ... 0110 0110	Line 9
0x 0120 0120 0120 0120 0120 ... 0120 0120	Line 10
0x 00E0 00E0 00E0 00E0 00E0 ... 00E0 00E0	Line 11
0x 00F0 00F0 00F0 00F0 00F0 ... 00F0 00F0	Line 12
0x 0100 0100 0100 0100 0100 ... 0100 0100	Line 13
:	:

Display(RGB565)

图 11.5: 测试模式

### 11.3.6 输入像素格式

DPI 支持多种像素格式输入，内部会将各种像素格式统一转成 RGB888 格式，再将 RGB888 格式转成 RGB565 格式发送出去。

### 11.3.6.1 YUV422(交织)

YUV422(交织) 格式输入数据的处理过程如下图所示:

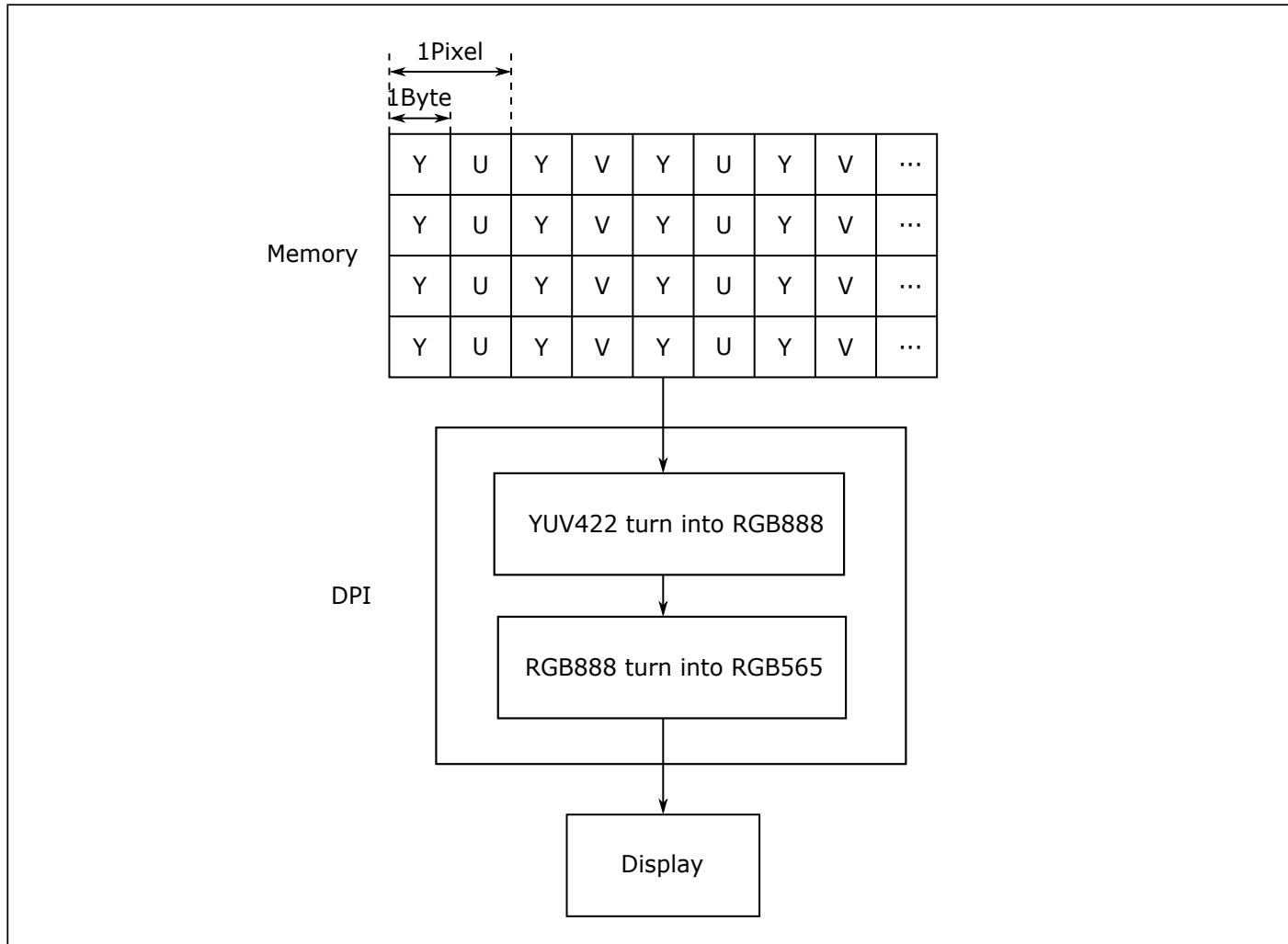


图 11.6: YUV422 处理过程

### 11.3.6.2 RGB888

RGB888 格式输入数据的处理过程如下图所示:

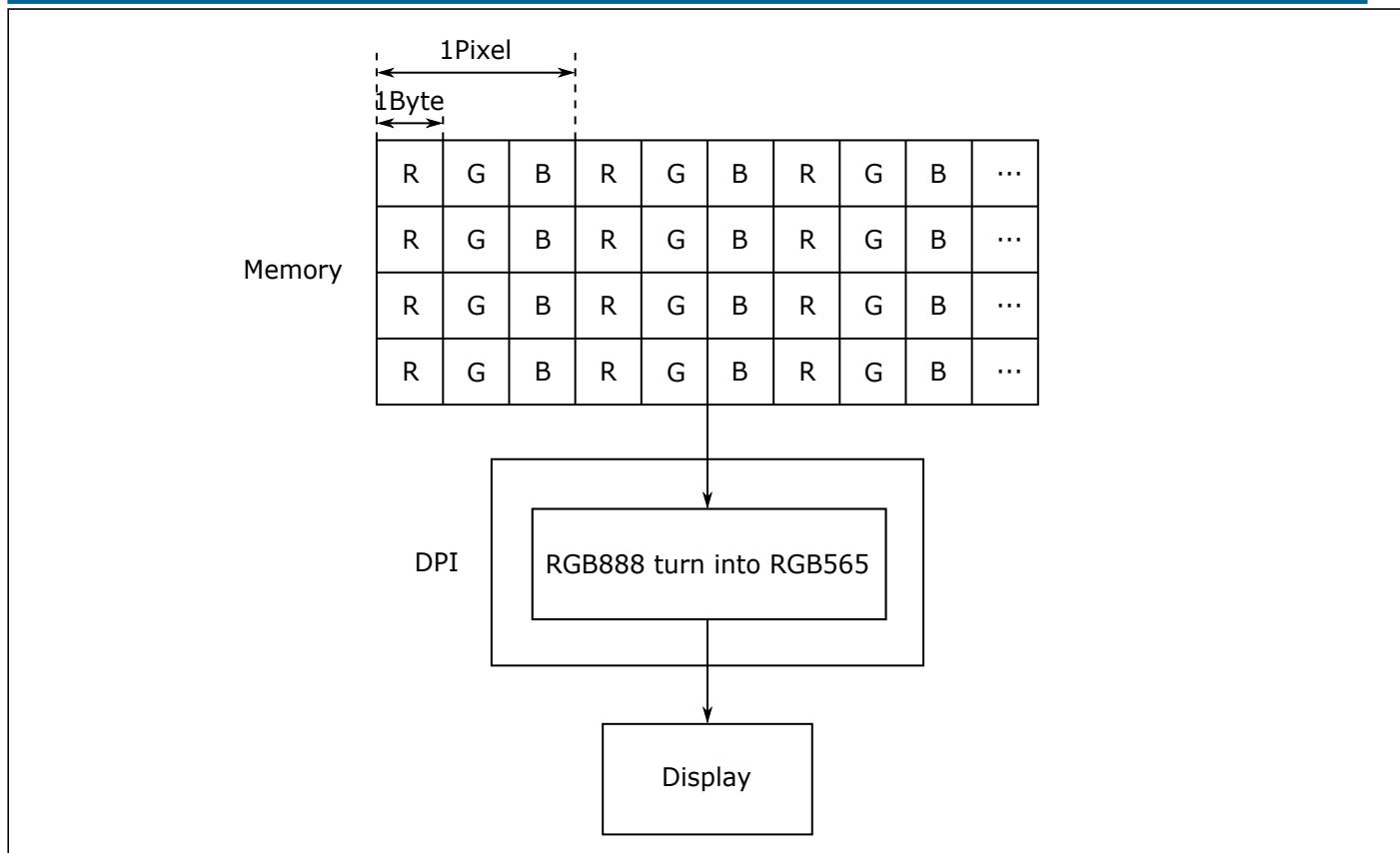


图 11.7: RGB888 处理过程

### 11.3.6.3 RGB565

RGB565 格式输入数据的处理过程如下图所示:

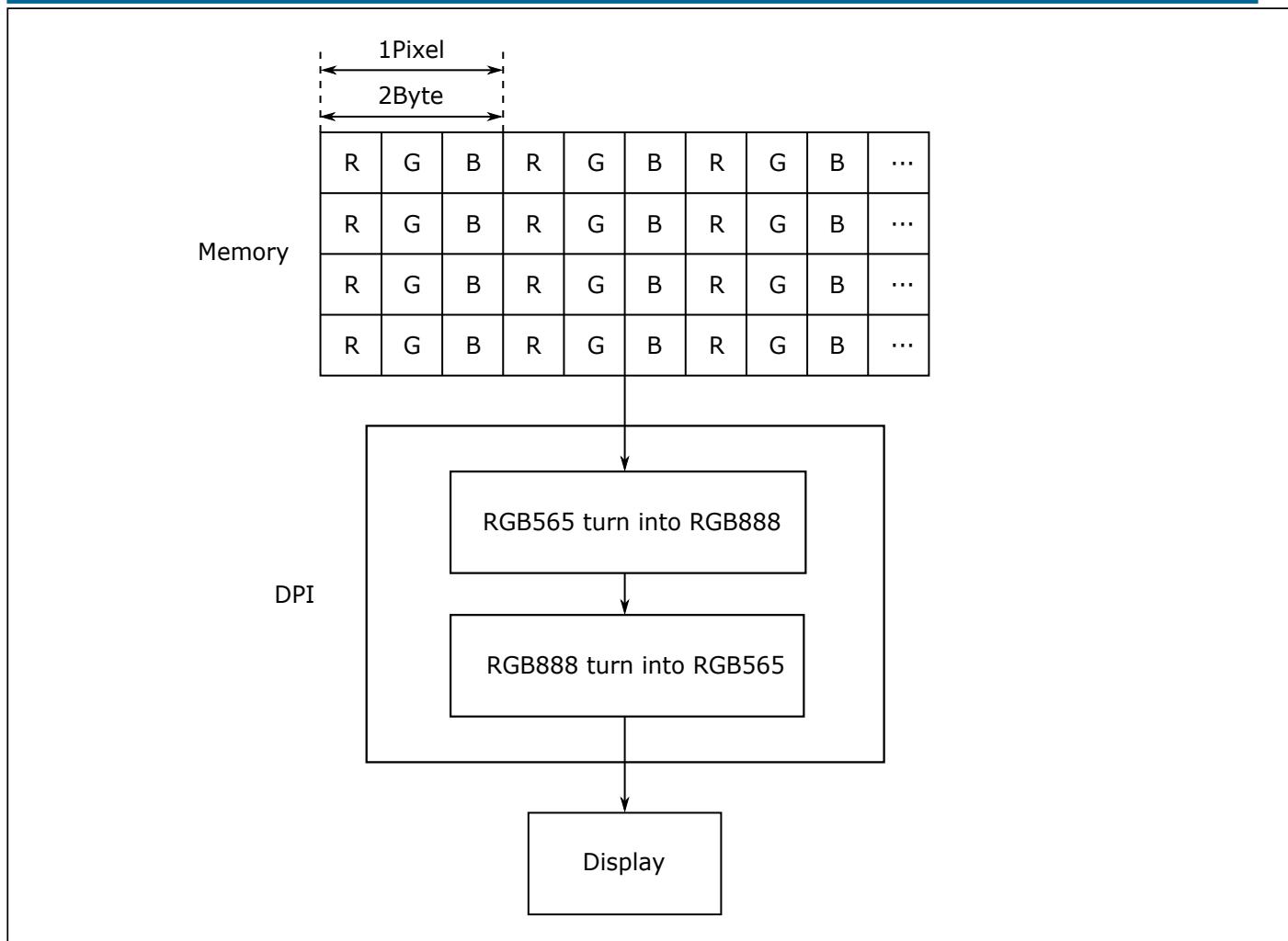


图 11.8: RGB565 处理过程

#### 11.3.6.4 RGBA8888

RGBA8888 格式输入数据的处理过程如下图所示：

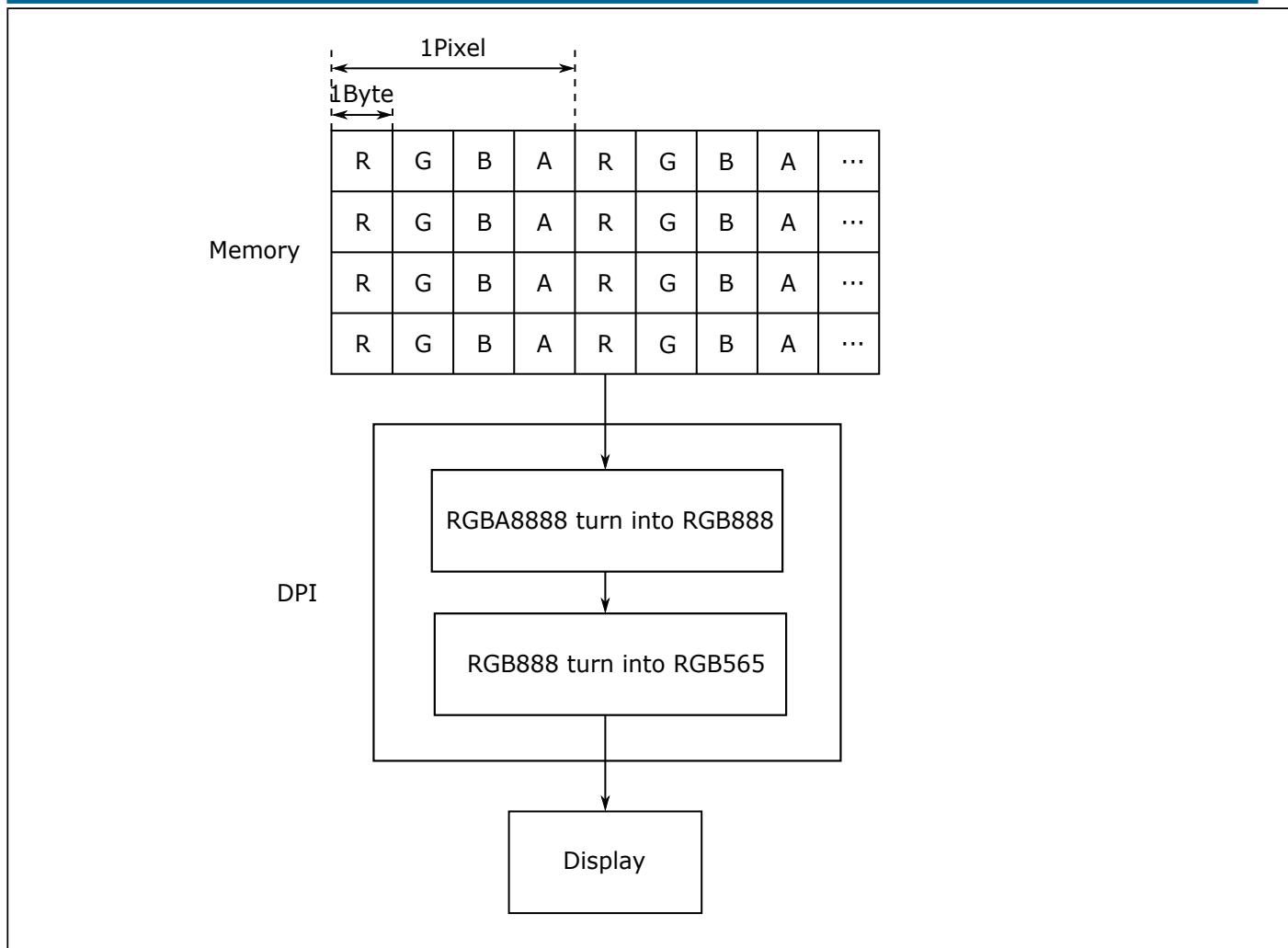


图 11.9: RGBA8888 处理过程

### 11.3.6.5 YUV420(平面)

YUV420(平面) 格式输入数据的处理过程如下图所示:

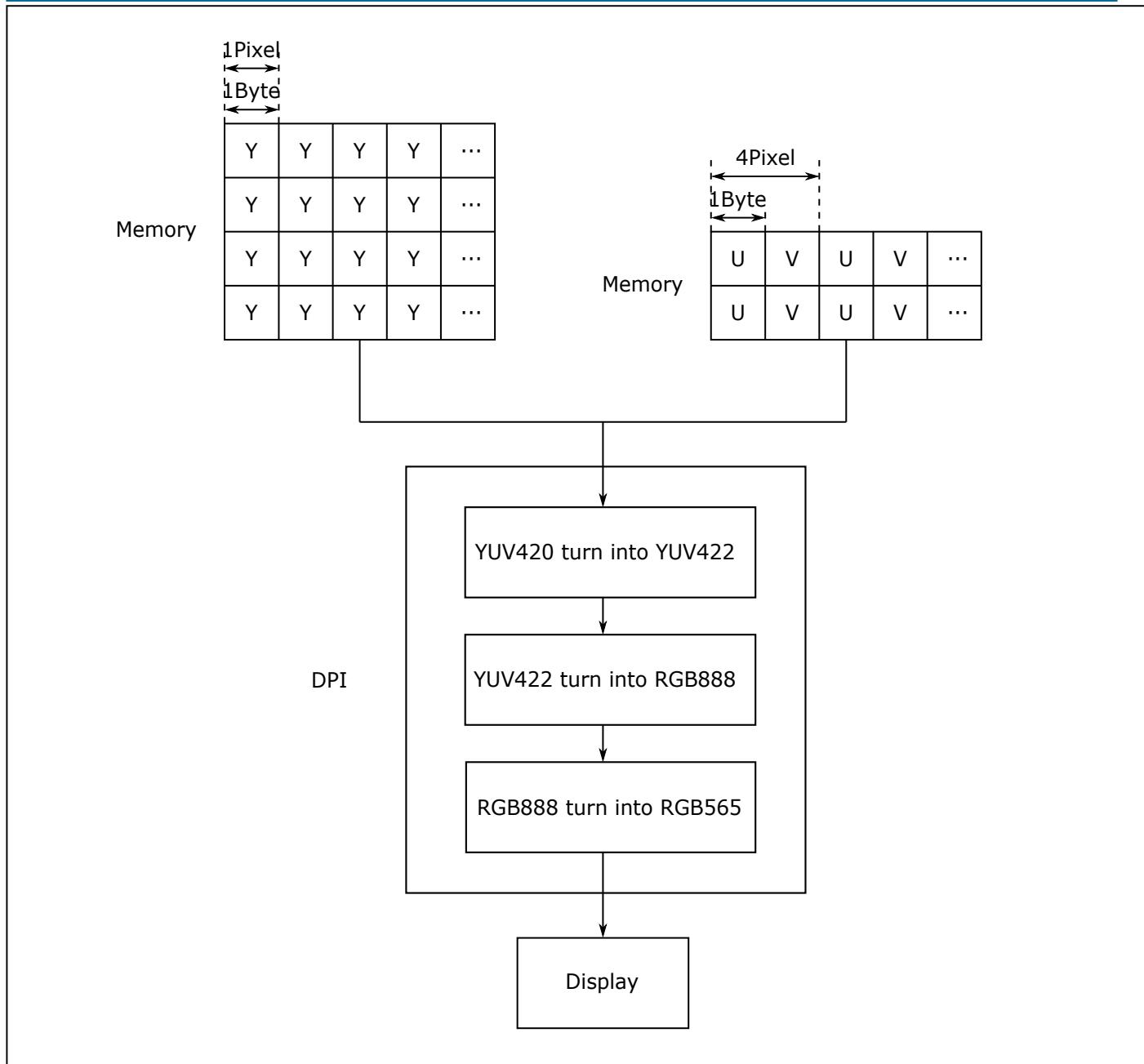


图 11.10: YUV420 处理过程

### 11.3.7 可编程中断

DPI 拥有一个可编程中断，通过配置，中断的触发源可以选择为 DPI 的输入 Vsync 或者输出 Vsync，跳变沿可以选择为上升沿或者下降沿。例如将中断配置为输入 Vsync 上升沿触发，则 DPI 会在其输入模块的每个 Vsync 上升沿产生一次中断请求；将中断配置为输出 Vsync 下降沿触发，则 DPI 会在其输出的每个 Vsync 下降沿产生一次中断请求。

### 11.3.8 与 OSD 配合使用

DSI 的数据输入可以配置为先经过 OSD 处理，OSD 的功能介绍请参阅 OSD 模块。

## 12.1 简介

DSI 是显示串行接口 (Display Serial Interface) 的简称，是 MIPI 联盟定义的用于与显示屏通信的接口协议。它向外设发送像素或命令，并可从外设读取状态或像素信息。DSI 在高速模式下所有的数据通道都是单向传输，低速模式下只有第一个数据通道是双向传输，其他通道为单向传输。时钟通道专用于在高速传输数据的过程中传输同步时钟信号。另外，一个主机端还允许同时与多个从属端进行通信。本芯片内置了 DSI 控制器和 D-PHY，可以实现与 DSI 兼容显示屏进行通信，从而将数据显示在屏幕上。

## 12.2 主要特点

- 符合 MIPI® 联盟标准
- MIPI® D-PHY 接口，最高速度可达 800Mbps
- 1 条 Clock Lane，最多支持 4 条 Data Lane
- Data Lane 支持四种排列顺序
- 支持 Ultra Low Power Mode 和 High Speed Mode
- Data Lane 0 支持双向通信和逃逸模式
- Escape 模式支持 LPDT/ULPS/Trigger
- LPDT 模式下支持长包和短包的发送
- 支持 ECC 和校验功能
- 支持数据长度错误和数据溢出错误中断
- 支持命令模式和视频模式
- 视频模式下支持 Event 和 Pulse 模式
- 显示接口支持的数据格式包括：
  - YUV422, 8-bit
  - RGB565
  - RGB666, loosely packed
  - RGB888
- 丰富多样的中断机制和状态查询
- LPDT 模式下支持 DMA 传输
- 可与 OSD 配合使用

## 12.3 功能描述

DSI 模块基本框图如下图所示：

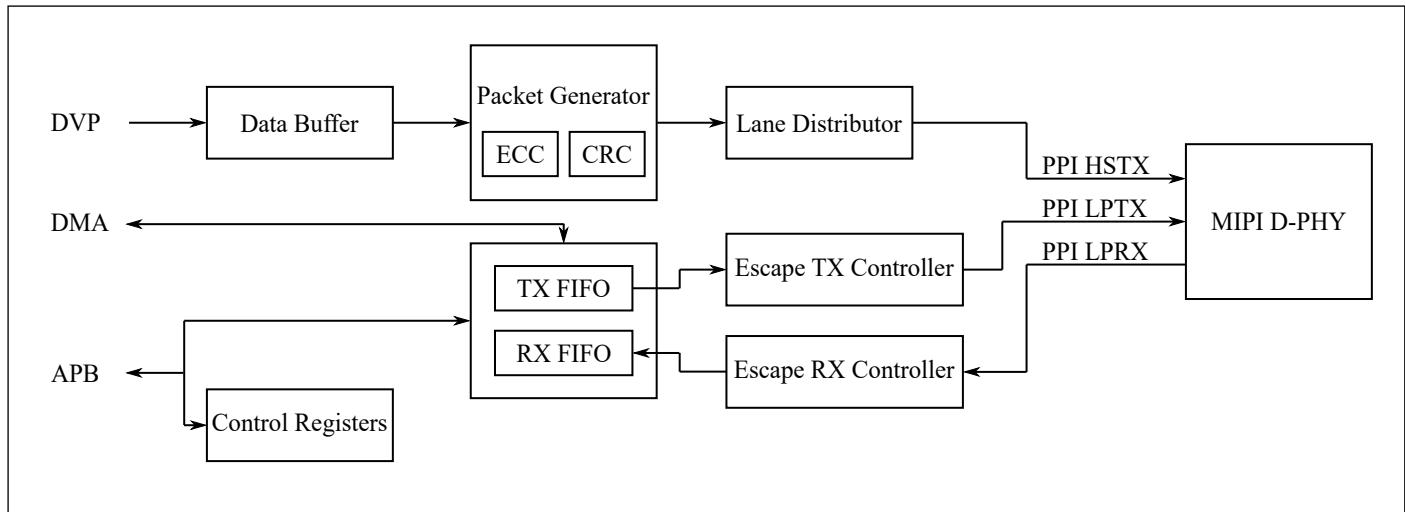


图 12.1: DSI 基本框图

### 12.3.1 DSI 协议分层

DSI 协议总共可分为 4 层，从下往上依次是物理层、通道管理层、协议层和应用层，如下图所示：

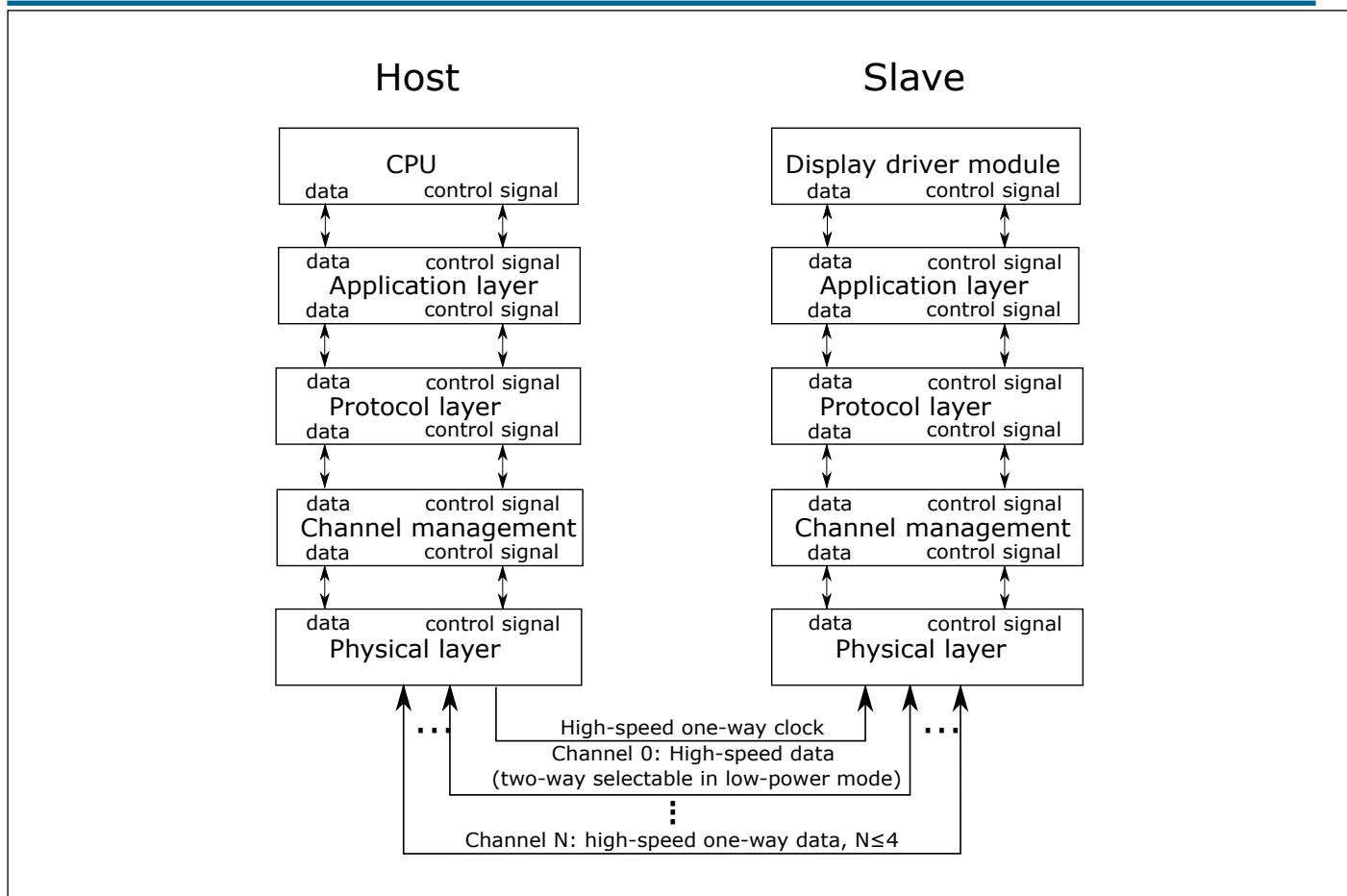


图 12.2: 协议分层

**物理层:** 物理层规定了传输介质(电导体)、输入/输出电路以及从串行位流捕获“1”和“0”的时钟机制。另外还规定了发送开始(SoT)和发送结束(EoT)的信令机制，以及可在发送和接收PHY之间传输的其他“带外”信息。

**通道管理层:** DSI 可扩展通道以提高性能，根据应用的带宽要求，数据通道的数量可以是 1、2、3 或 4。通道管理层的作用是在发送端将需要发送的数据按照通道次序分组输送到相应的数据通道，在接收端从各个通道收集数据后将它们合并到一个重新组合的数据流中，从而恢复出原始流序列。

**协议层:** DSI 协议定义了数据包格式，包括短数据包和长数据包。协议层的作用是在发送端根据数据的类型和内容进行组包，完成ECC 码和 CRC 码的添加并传输到通道管理层；在接收端根据 ECC 码和 CRC 码对收到的数据包进行检错纠错，完成对包头和数据内容的译码并传输到应用层。

**应用层:** 根据应用模块的需要，应用层在发送端对要发送的命令和数据进行初步编码转化为 DSI 所规定的格式，在接收端将接收的数据还原为应用模块支持的数据格式和时序要求。

## 12.3.2 物理层

按照 D-PHY 协议，物理层中在主机端和从属端之间采用的是同步连接，时钟通道 (Clock Lane) 用于传送高速时钟，一个或多个数据通道 (Data Lanes) 用于传送低功耗数据信号或高速数据信号。每一个通道都是利用两根互连线实现主机端和从属端的连接，且都支持高速 (HS) 模式和低功耗 (LP) 模式。在高速模式下，发送端同时驱动该通道的两根互连线，输出低摆幅差分信号，例如 200mV；在低速模式下，发送端会分别驱动互连线，各自输出摆幅相对较大的单端信号，例如 1.2V。两种模式下的互连线电平如下图所示：

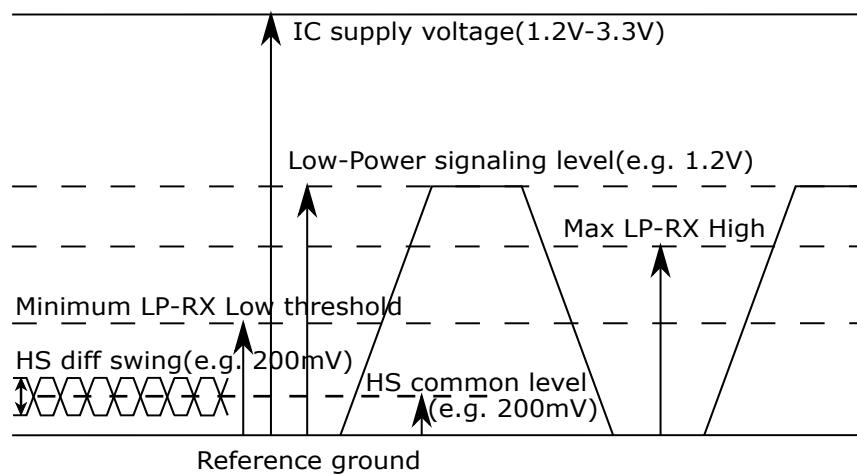


图 12.3: 高速和低功耗模式下的电平

互连线两端是驱动单元和接收单元。驱动单元包括差分发送模块 (HS\_TX) 和低功耗单端发送模块 (LP\_TX)，接收单元包括差分接收模块 (HS\_RX) 和低功耗单端接收模块 (LP\_RX)。差分发送模块 (HS\_TX) 以差分信号驱动互连线，高速通道上呈现两种状态：Differential-0 和 Differential-1。低功耗单端发送模块 (LP\_TX) 独立地驱动两根互连线，通道上有四种不同的状态：LP-00、LP-01、LP-10 和 LP-11。协议针对线路电平作了具体的定义并设置了三种操作模式：高速模式、控制模式和 Escape 模式，如下表所示：

Lane Pair State Code	Line DC Voltage Levels		High Speed(HS)	Low Power	
	DATA_P	DATA_N	Burst Mode	Control Mode	Escape Mode
HS-0	Low(HS)	High(HS)	Differential-0	Note 1	Note 1
HS-1	High(HS)	Low(HS)	Differential-1	Note 1	Note 1
LP-00	Low(LP)	Low(LP)	Not Defined	Bridge	Space
LP-01	Low(LP)	High(LP)	Not Defined	HS-Request	Mark-0
LP-10	High(LP)	Low(LP)	Not Defined	LP-Request	Mark-1
LP-11	High(LP)	High(LP)	Not Defined	Stop	Note 2

Note:

- When the lane is in high-speed mode, the low-power receiver (LP\_Rx) of each lane will check the LP-00 status code;
- If the low power receiver (LP-Rx) of each lane recognizes the LP-11 status code, the lane will return to control mode.

图 12.4: 通道状态码

### 12.3.3 时钟通道 (Clock Lane)

时钟通道可以被驱动到三种不同的功耗模式: 低功耗模式 (Low Power Mode, LPM)、超低功耗模式 (Ultra-Low Power Mode, ULPM) 和高速时钟模式 (High Speed Clock Mode, HSCM)。时钟通道不同功耗模式相互切换的主要流程图如下所示:

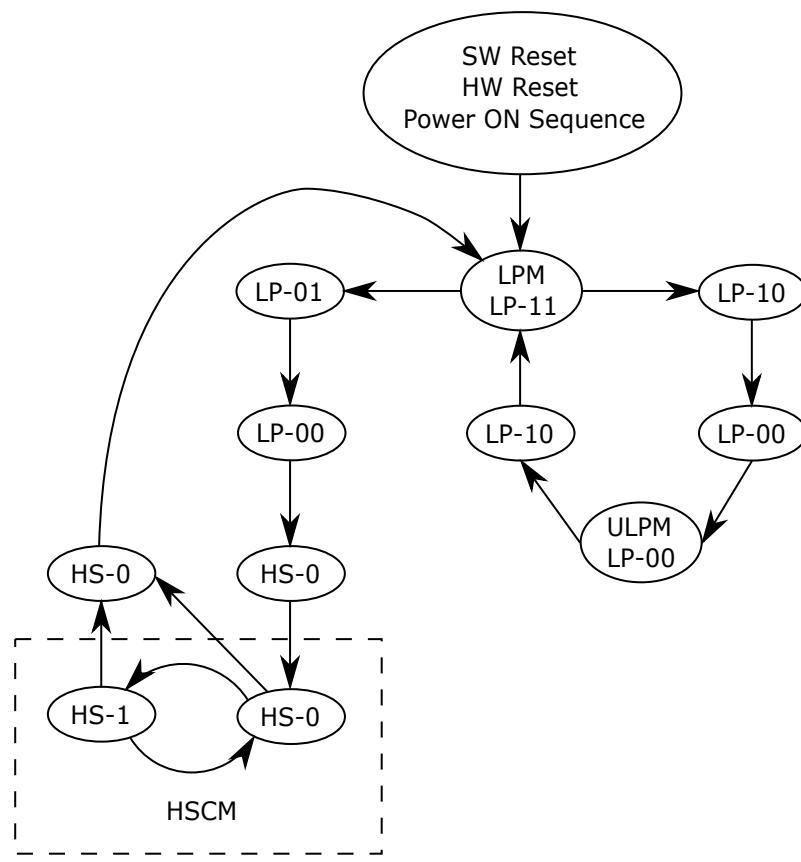


图 12.5: 时钟通道模式切换图

### 12.3.3.1 低功耗模式 (LPM, LP-11)

有三种方式进入:

1. 在软件复位、硬件复位和上电时序后，自动进入 LP-11(LPM)。
2. 离开超低功耗模式 (ULPM, LP-00) 后先变为 LP-10，再进入 LP-11(LPM)，如下图所示：

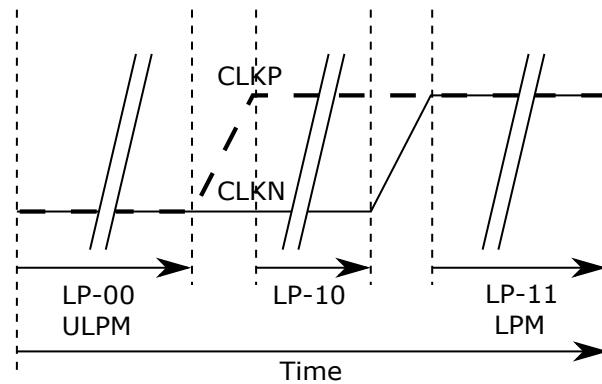


图 12.6: 由超低功耗模式进入低功耗模式

3. 离开高速时钟模式 (HSCM, HS-0 或 HS-1) 后先变为 HS-0, 再进入 LP-11(LPM), 如下图所示:

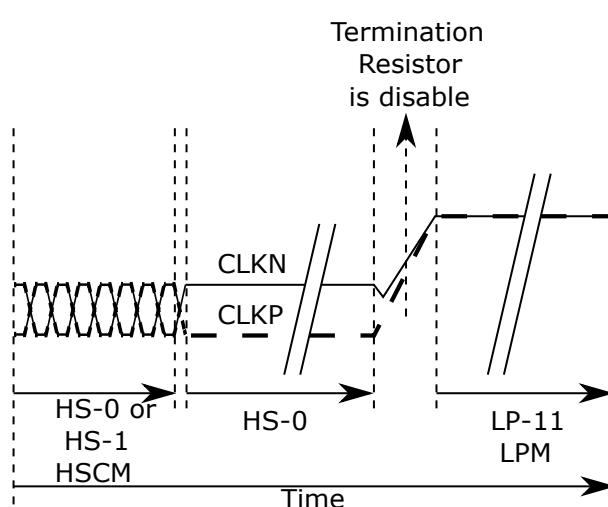


图 12.7: 由高速时钟模式进入低功耗模式

### 12.3.3.2 超低功耗模式 (ULPM, LP-00)

只有一种方式进入: 从低功耗模式 (LPM, LP-11) 先变为 LP-10, 再进入 LP-00(ULPM), 如下图所示:

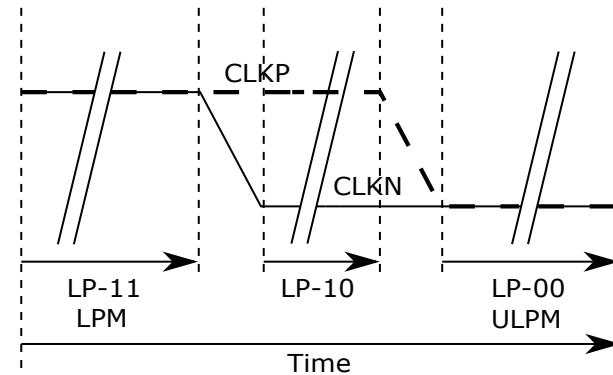


图 12.8: 由低功耗模式进入超低功耗模式

### 12.3.3.3 高速时钟模式 (HSCM, HS-0/1)

只有一种方式进入：从低功耗模式 (LPM, LP-11) 先变为 LP-01，再变为 LP-00，然后变为 HS-0，之后进入 HS-0/1(HSCM)，如下图所示：

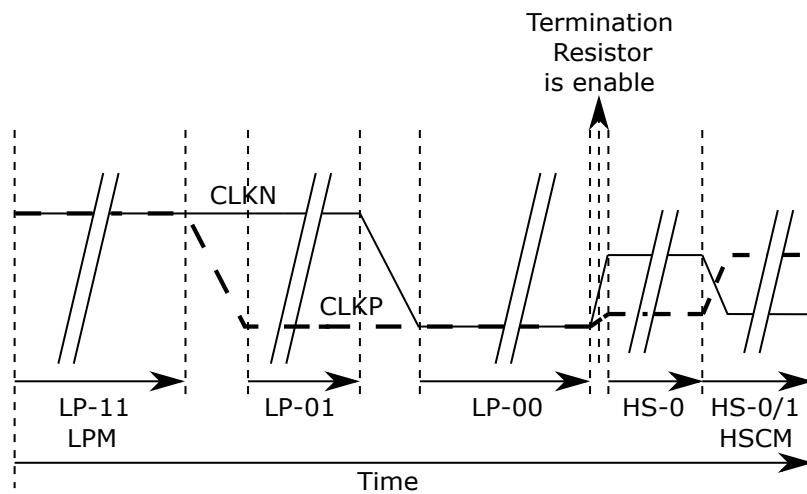


图 12.9: 由低功耗模式进入高速时钟模式

### 12.3.4 数据通道 (Data Lanes)

所有的数据通道 (D3P/N、D2P/N、D1P/N 和 D0P/N) 都可以被驱动到高速数据传输 (High-Speed Data Transmission, HSDT) 模式，但只有数据通道 0(D0P/N) 可以进入 Escape 模式和 Bus Turn-Around Request(BTA) 模式。三种模式的进入和退出序列如下表所示：

Mode	Entering Mode Sequence	Leaving Mode Sequence
Escape Mode	LP-11 -> LP-10 -> LP-00 -> LP-01 -> LP-00	LP-00 -> LP-10 -> LP-11(Mark-1)
High-Speed Data Transmission	LP-11 -> LP-01 -> LP-00 -> HS-0	(HS-0 or HS-1) -> LP-11
Bus Turnaround Request	LP-11 -> LP-10 -> LP-00 -> LP-10 -> LP-00	Hi-Z

图 12.10: 数据通道三种模式进入和退出序列

#### 12.3.4.1 高速数据传输 (HSDT)

当 MCU 的时钟通道已经进入高速时钟模式 (HSCM) 后显示模块可以进入高速数据传输 (HSDT)。进入高速数据传输的序列如下图所示：

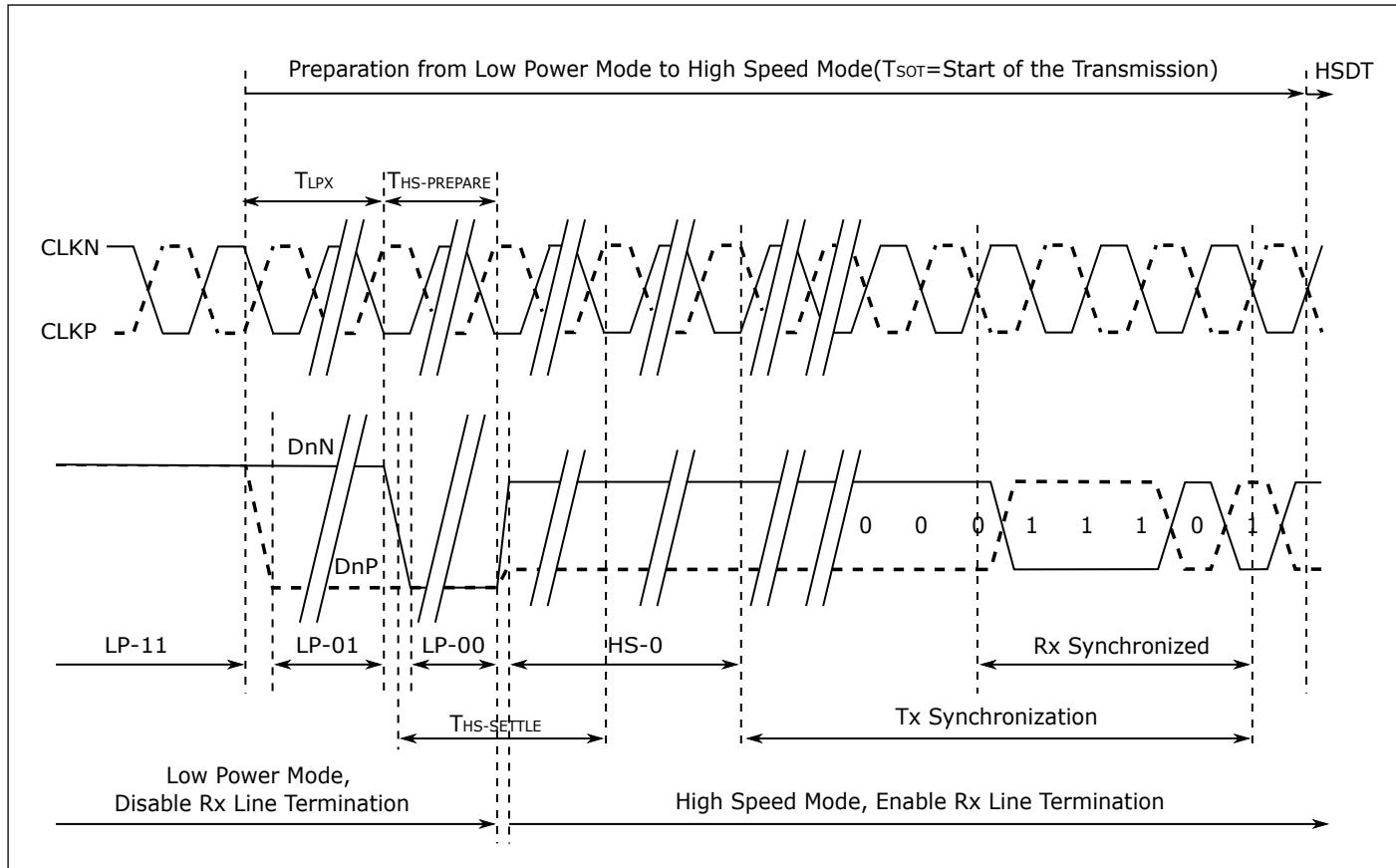


图 12.11: 高速数据传输进入序列

序列:

- 开始: LP-11;
- HS 请求: LP-01;
- HS 确定: LP-00=>HS-0(接收端:Lane 终端使能)
- 接收端同步: 011101(按发送的 bit 顺序), 发送端同步: 00011101(按发送的 bit 顺序);
- 结束: 高速数据传输, 准备好接收加载的高速数据。

当 MCU 的时钟通道处于高速时钟模式 (HSCM) 时显示模块可以退出高速数据传输, 并且 HSCM 必须保持到所有的数据通道都进入 LP-11 模式。退出高速数据传输的序列如下图所示:

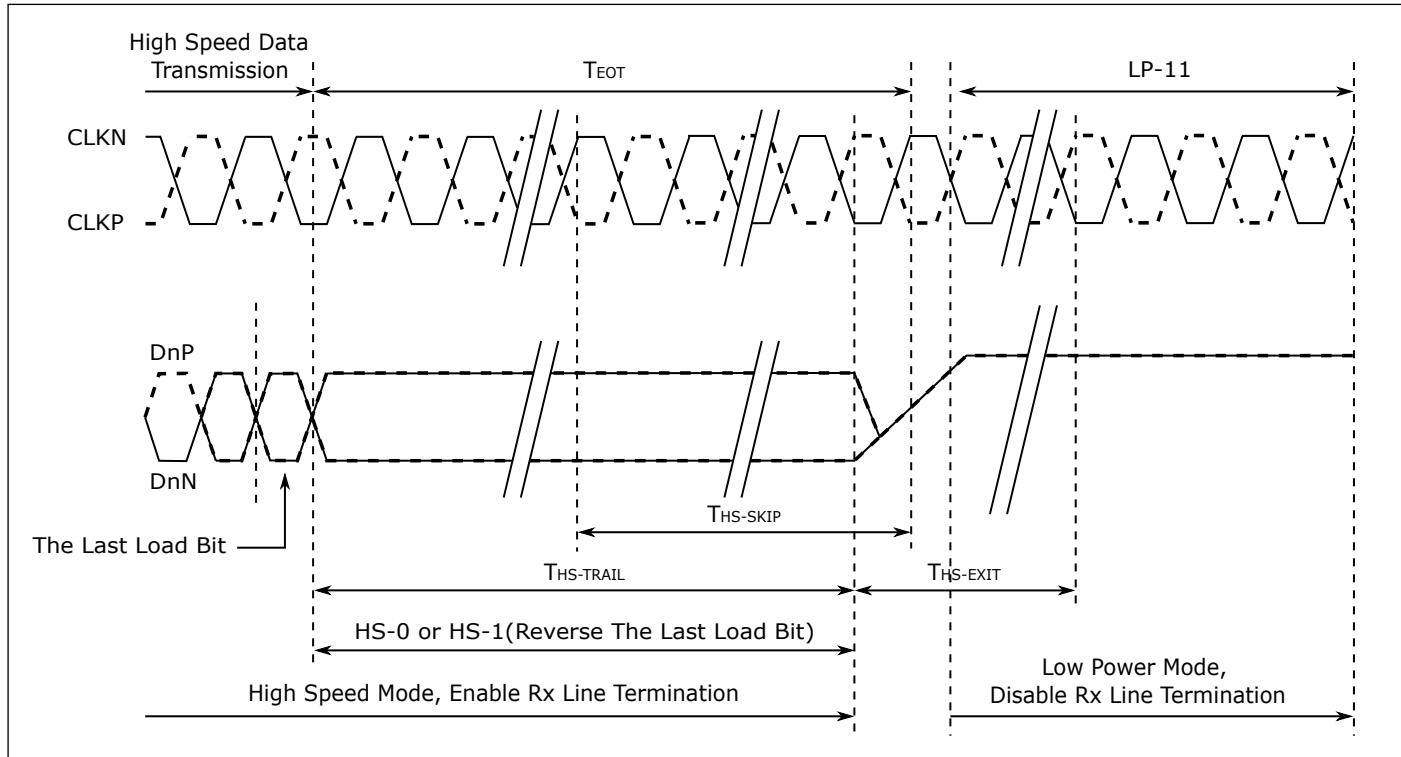


图 12.12: 高速数据传输退出序列

序列:

- 开始: 高速数据传输;
- 停止高速数据传输: 如果最后的加载位是 HS-0 则 MCU 变为 HS-1, 如果最后的加载位是 HS-1 则 MCU 变为 HS-0;
- 结束: LP-11(接收端: Lane 终端禁用)。

高速数据传输的突发操作可以由一个数据包或者多个数据包组成, 这些数据包可以是长包也可以是短包。以下是一些不同的高速数据传输突发例子:

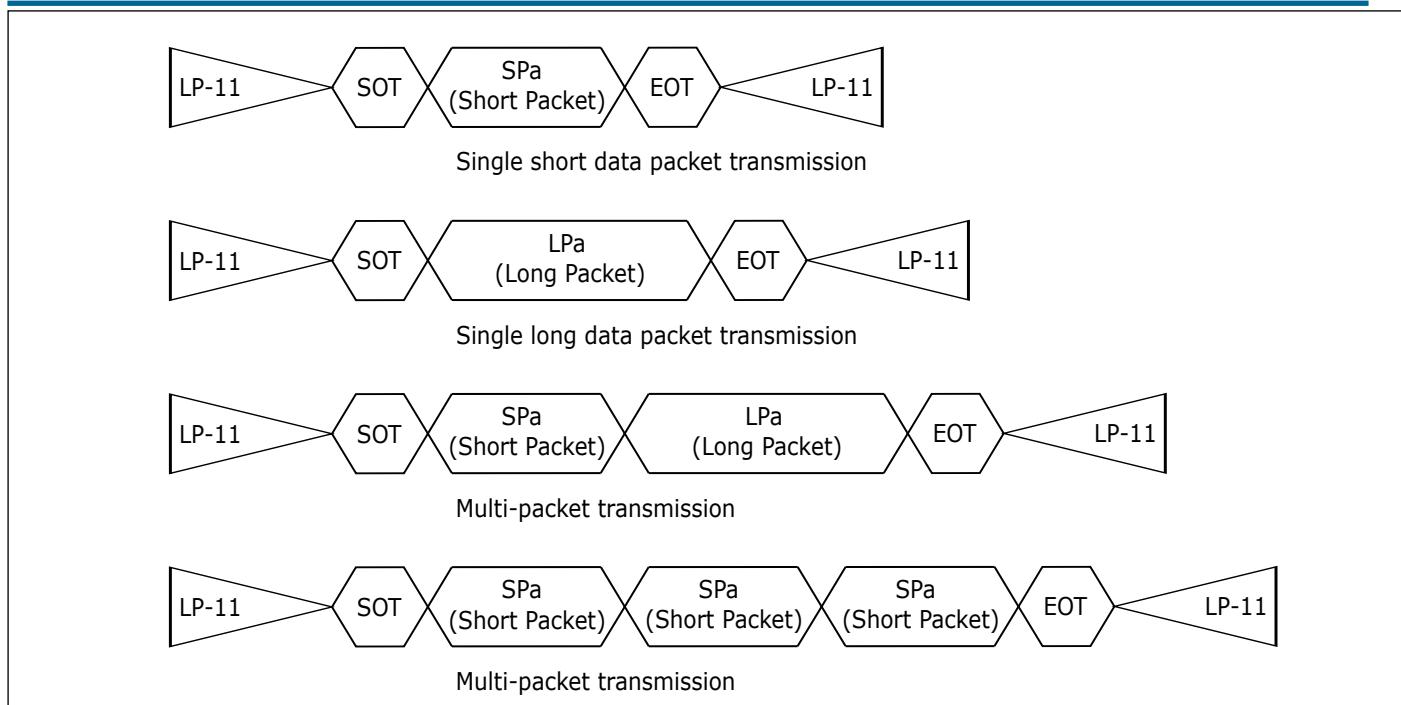


图 12.13: 高速数据传输突发序列

#### 12.3.4.2 Bus Turn-Around(BTA)

当正在控制通道 DOP/N 的 MCU 或显示模块想要从接收端获取信息时，可以开启一个总线反转过程，使总线控制权发生反转。MCU 和显示模块发起总线反转过程使用相同的序列，如下图所示：

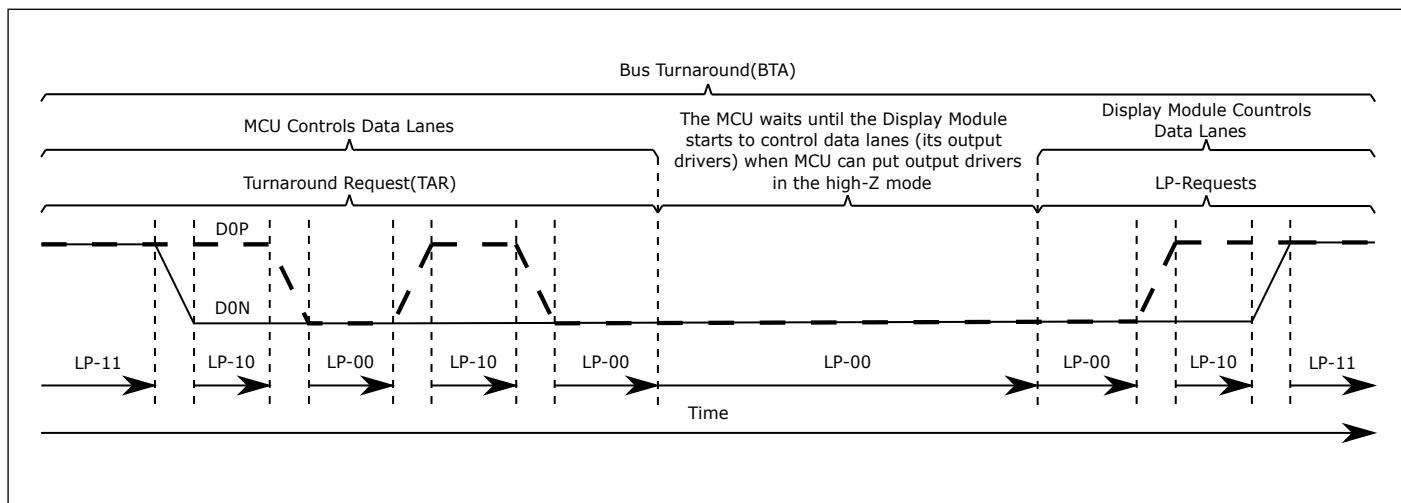


图 12.14: 总线反转序列

序列：

- 开始：LP-11；
- 反转请求 (MCU): LP-11=>LP-10=>LP-00=>LP-10=>LP-00；

- MCU 会等待直到显示模块开始控制通道 D0P/N，而 MCU 则停止控制通道 D0P/N 并设为高阻态；
- 显示模块变为停止模式：LP-00=>LP-10=>LP-11。

### 12.3.4.3 Escape 模式

Escape 模式是 LPM 下的一种特殊模式，有以下几个作用：

- 1. 从 MCU 发送低功耗传输 (LPDT) 命令到显示模块；
- 2. 驱动数据通道到超低功耗状态 (ULPS)；
- 3. 发送远程复位 (RAR) 命令，复位显示模块；
- 4. 发送应答 (ACK) 信号，用于从显示模块传输非错误事件到 MCU。

Escape 模式的基本序列如下图所示：

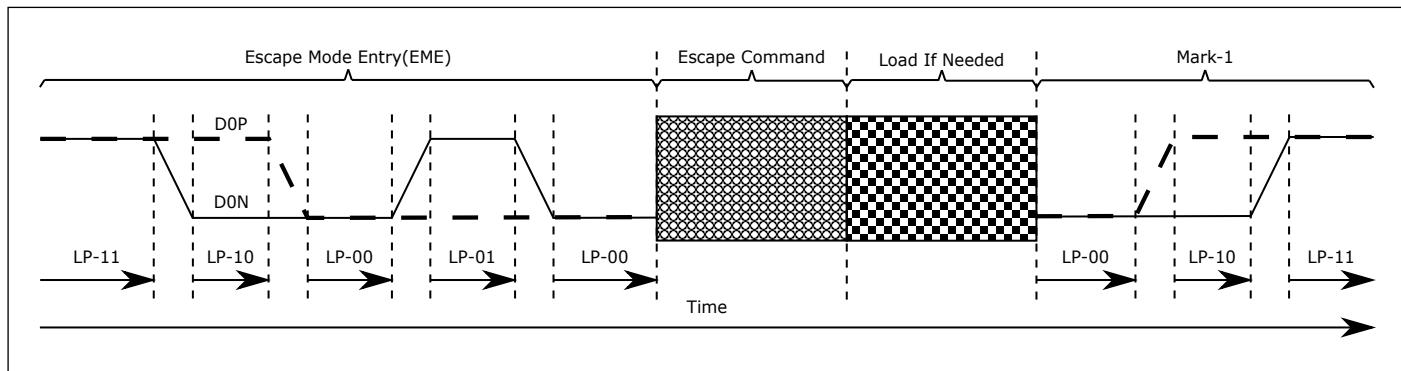


图 12.15: Escape 模式序列

序列：

- 开始：LP-11；
- Escape 模式进入 (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00；
- Escape 命令 (EC): 当其中一个数据通道从低变高再变低时，此变化显示了当前数据位的值 (D0P = 1, D0N = 0)。
- 当数据通道 0 从低变高再变低，接收器会锁存 1bit 数据，即逻辑 0。接收器会把这种从低变高再变低的传输作为其内部的时钟；
- 如果需要的话，加载数据；
- 退出 Escape: LP-00=>LP-10=>LP-11；
- 结束: LP-11。

Escape 命令种类如下表所示：

Escape Command	Command Type	Entry Command Pattern (First Bit->Last Bit Transmitted)	Dn	D0
Low-Power Data Transmission	Mode	1110 0001 b	✗	✓
Ultra-Low Power Mode	Mode	0001 1110 b	✓	✓
Note 1	Mode	1001 1111 b	✗	✗
Note 1	Mode	1101 1110 b	✗	✗
Remote Application Reset	Trigger	0110 0010 b	✗	✓
Tearing Effect	Trigger	0101 1101 b	✗	✗
Acknowledge	Trigger	0010 0001 b	✗	✓
Note 1	Trigger	1010 0000 b	✗	✗

Note:

1. The support of this command has not been implemented on the display module;
2. The value of n is 1, 2 and 3;
3. "✓" means support;
4. "✗" means not supported.

图 12.16: Escape 模式命令

#### 12.3.4.4 低功耗数据传输 (LPDT)

当数据通道进入 **Escape** 模式，并且 MCU 已经将 LPDT 命令发送给显示模块，则 MCU 可以在低功耗模式下将数据发送到显示模块。显示模块发送数据到 MCU 时也使用同样的序列。LPDT 序列如下图所示：

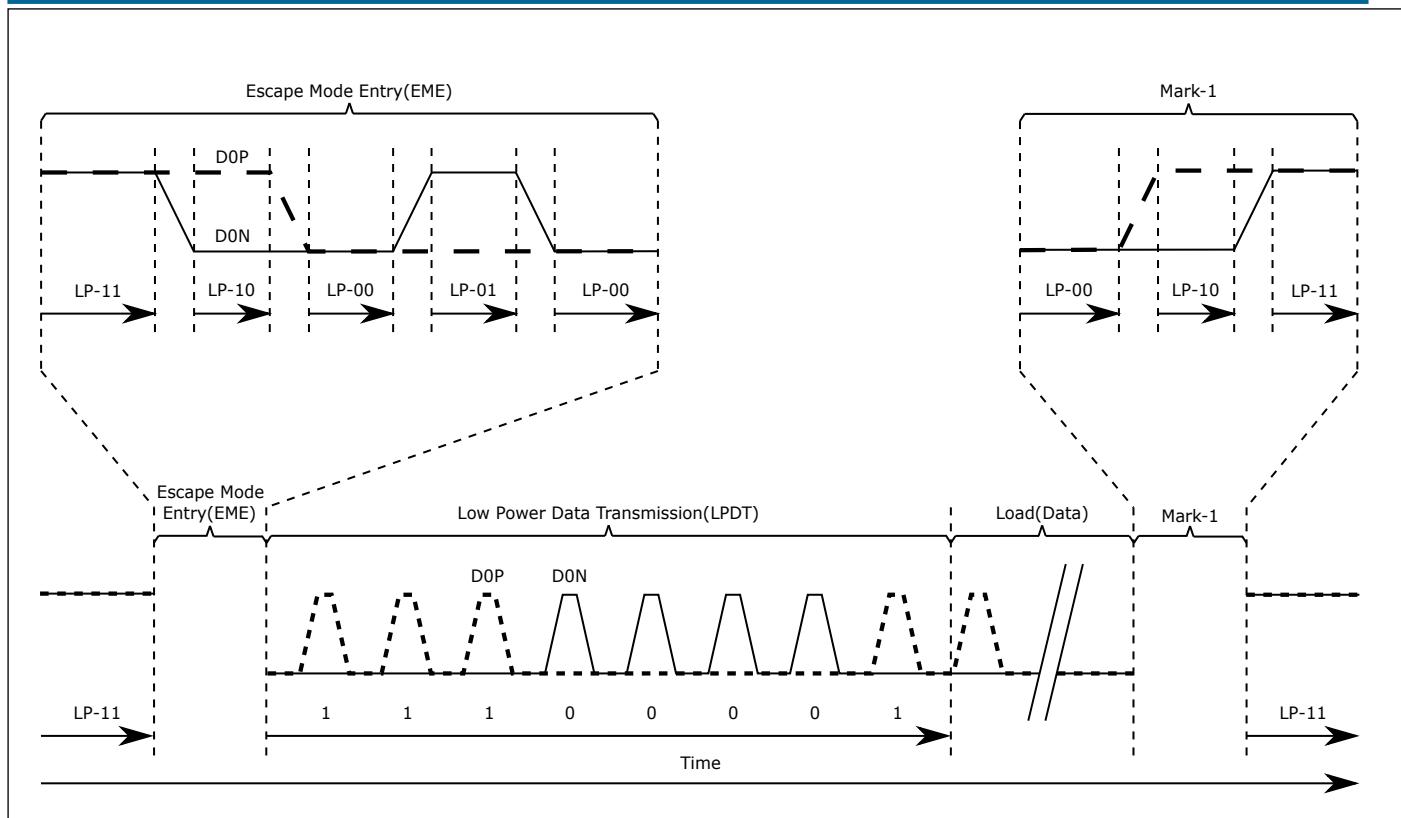


图 12.17: 低功耗数据传输序列

序列:

- 开始: LP-11;
- Escape 模式进入 (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00;
- Escape 模式下发送 LPDT 命令: 0x87(LSB-first);
- 加载数据: 一个或多个字节, 当数据通道在字节之间停止 (所有通道都为低) 时, 处于暂停状态;
- 退出 Escape: LP-00=>LP-10=>LP-11;
- 结束: LP-11。

### 12.3.4.5 超低功耗状态 (ULPS)

当数据通道进入 Escape 模式时，MCU 可以强制数据通道进入超低功耗状态。ULPS 序列如下图所示：

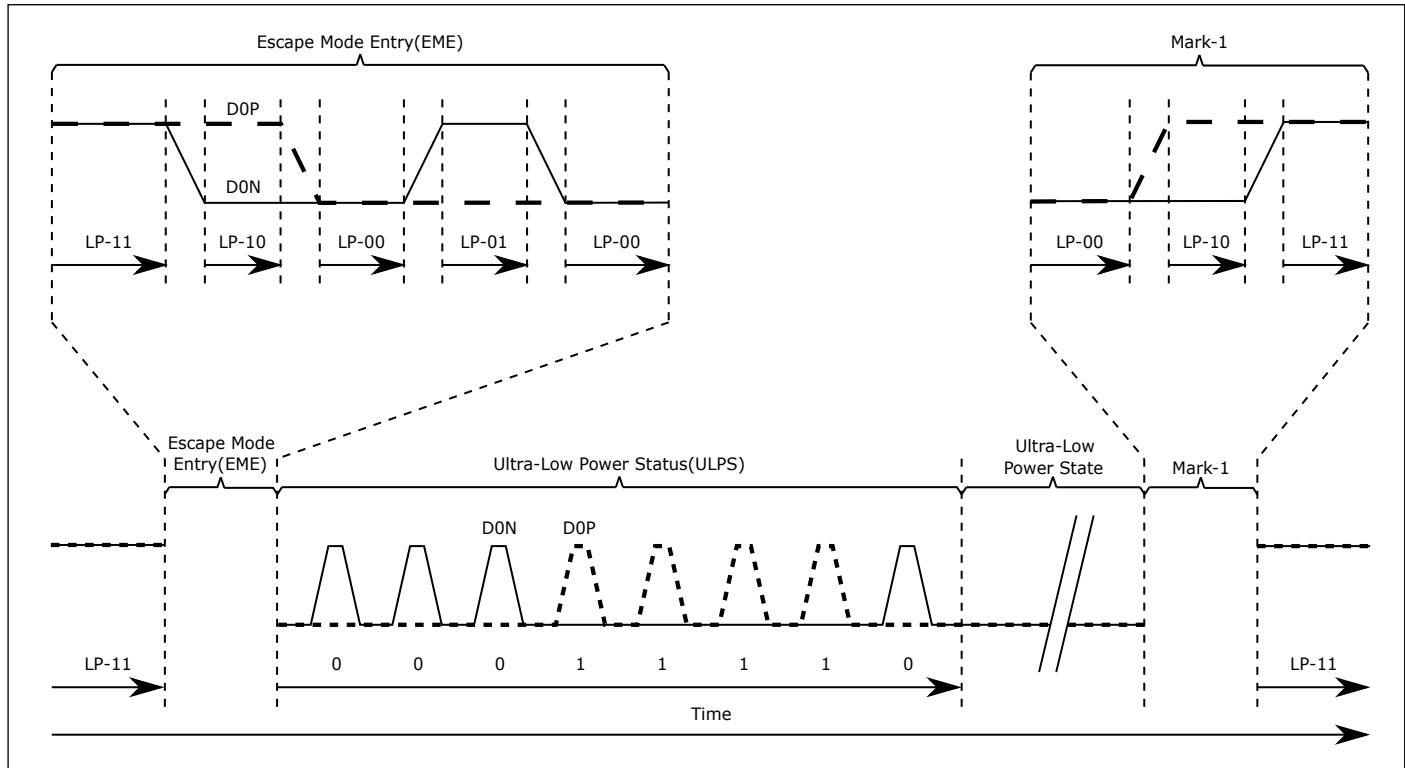


图 12.18: 超低功耗状态序列

序列：

- 开始：LP-11；
- Escape 模式进入 (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00;
- Escape 模式下发送 ULPS 命令：0x78(LSB-first)；
- 超低功耗状态，MCU 保持数据通道为低；
- 退出 Escape: LP-00=>LP-10=>LP-11；
- 结束: LP-11(数据通道退出 ULPS 之后必须等待 100us 才能发送下一条命令)。

### 12.3.4.6 远程复位 (RAR)

远程复位是 Trigger 命令中的一种，当数据通道进入 Escape 模式时，MCU 可以通知显示模块在远程申请复位触发器中进行复位。RAR 序列如下图所示：

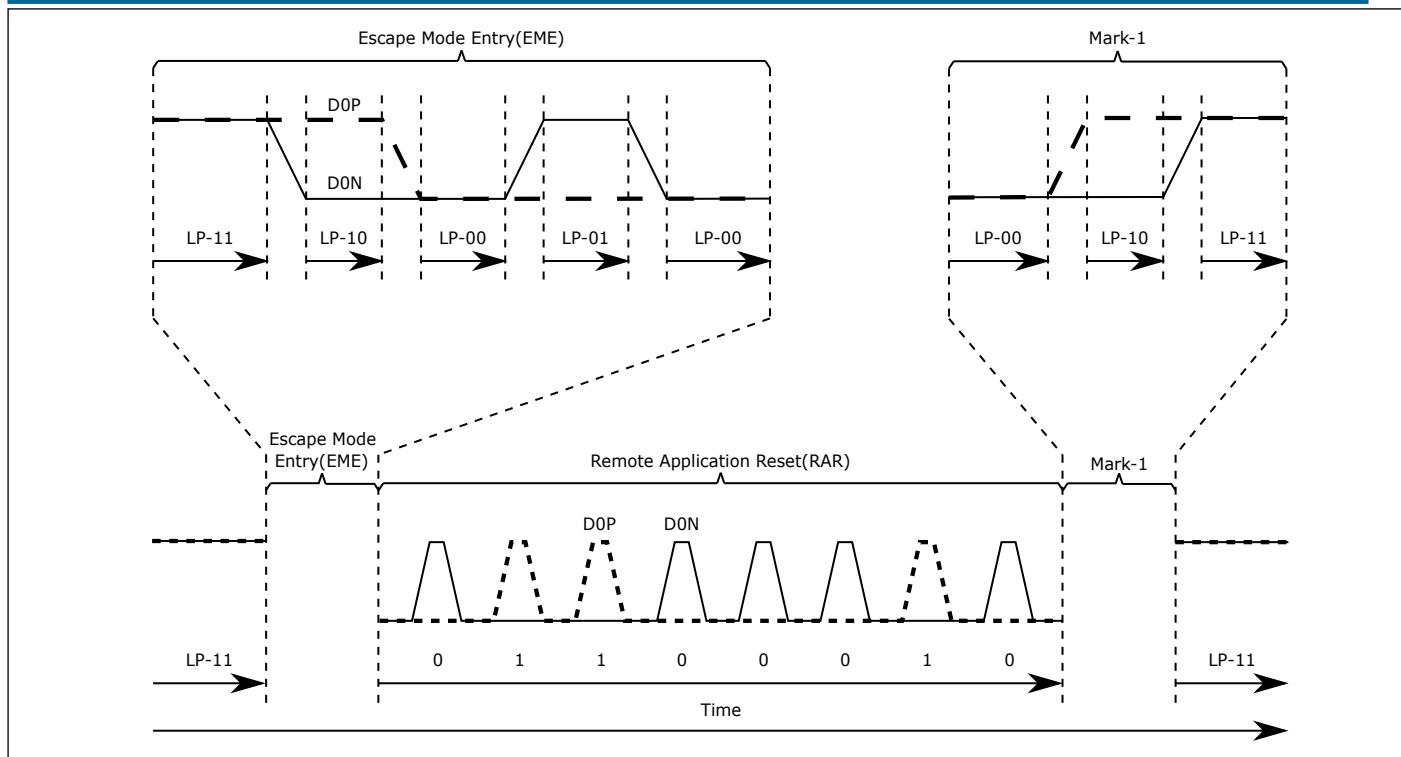


图 12.19: 远程复位序列

序列:

- 开始: LP-11;
- Escape 模式进入 (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00;
- Escape 模式下发送 RAR 命令: 0x46(LSB-first);
- 退出 Escape: LP-00=>LP-10=>LP-11;
- 结束: LP-11。

#### 12.3.4.7 应答 (ACK)

应答是 Trigger 命令中的一种，显示模块可以通过 ACK 通知 MCU 有未识别的错误。显示模块发送 ACK 的序列如下图所示：

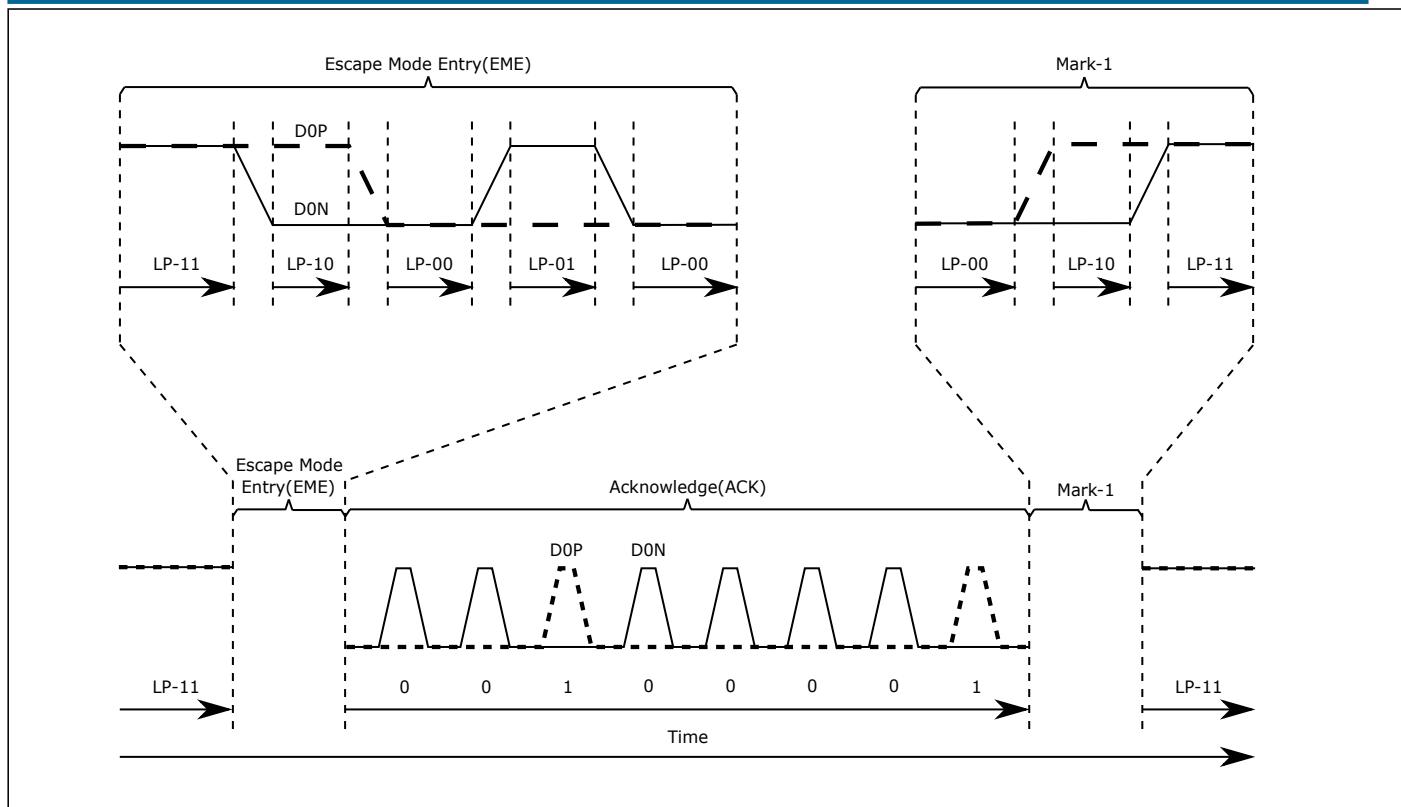


图 12.20: 应答序列

序列：

- 开始：LP-11；
- Escape 模式进入 (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00;
- Escape 模式下发送 ACK 命令: 0x84(LSB-first);
- 退出 Escape: LP-00=>LP-10=>LP-11;
- 结束: LP-11

### 12.3.5 通道管理层

DSI 是一种通道可扩展接口，按照实际应用的带宽需求可能会将数据通道从一条扩展为两条、三条或者四条，并获得峰值总线带宽的近似线性增加。在发射端，通道管理层的作用是把协议层的数据按照一定的顺序分发到每个数据通道上；在接收端，通道管理层的作用是从各个通道收集传输的字节，并将字节合并成完整的数据包。所有的数据通道共用一个时钟信号，但每个通道不一定是同时传输完成的，当传输的字节数不是通道数的整数倍时，某些通道会在其他通道之前传完数据。除通道 0 之外，其他三条通道的排列顺序可以通过 `<CR_LANE_MUX_SEL>` 配置。

有以下四种排列顺序：

- Lane0,Lane1,Lane2,Lane3
- Lane0,Lane3,Lane1,Lane2
- Lane0,Lane2,Lane3,Lane1
- Lane0,Lane2,Lane1,Lane3

分别以单通道和四通道 (顺序为 Lane0,Lane1,Lane2,Lane3) 为例, 发送端如下图所示:

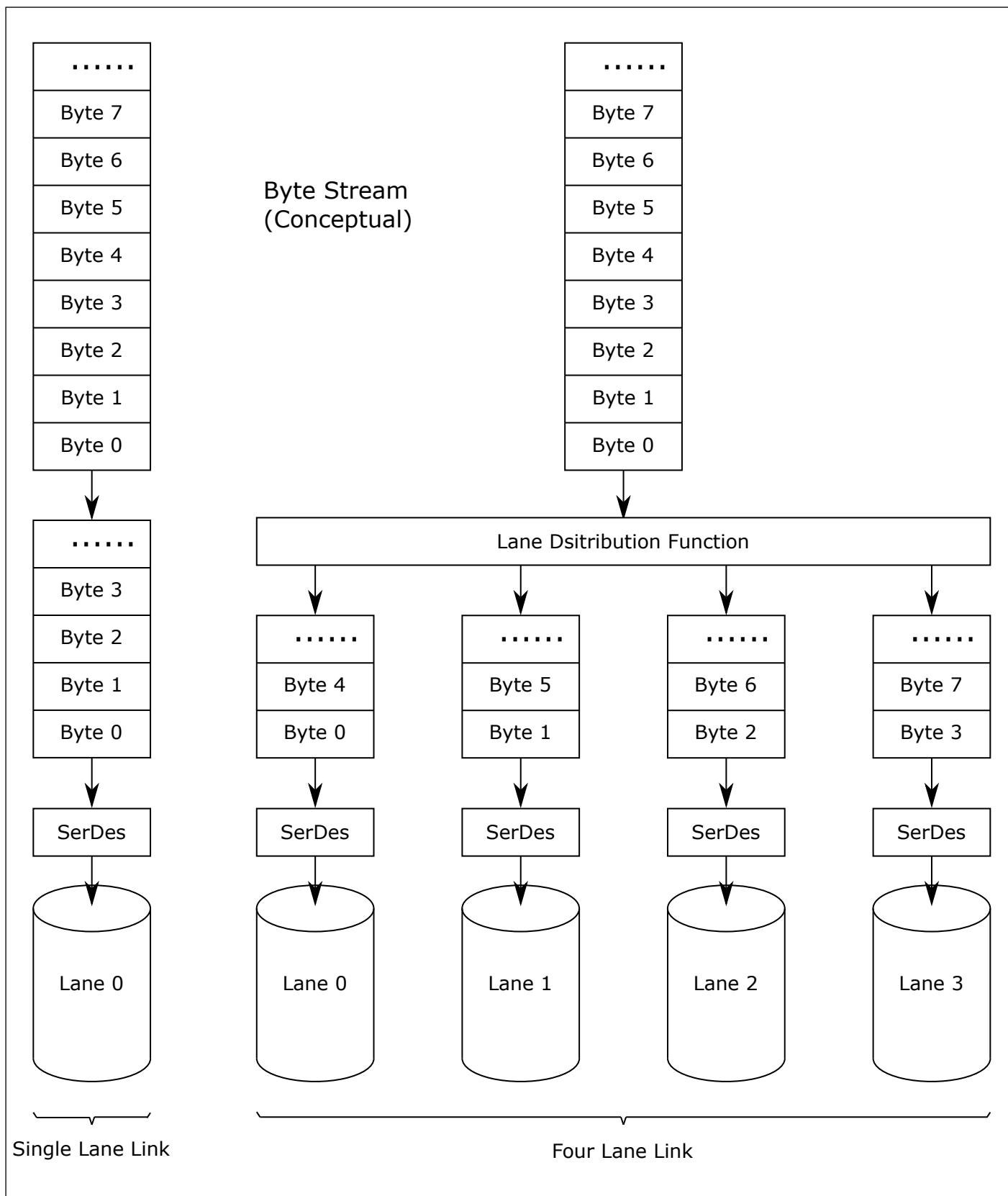


图 12.21: 发送端拆分

接收端如下图所示：

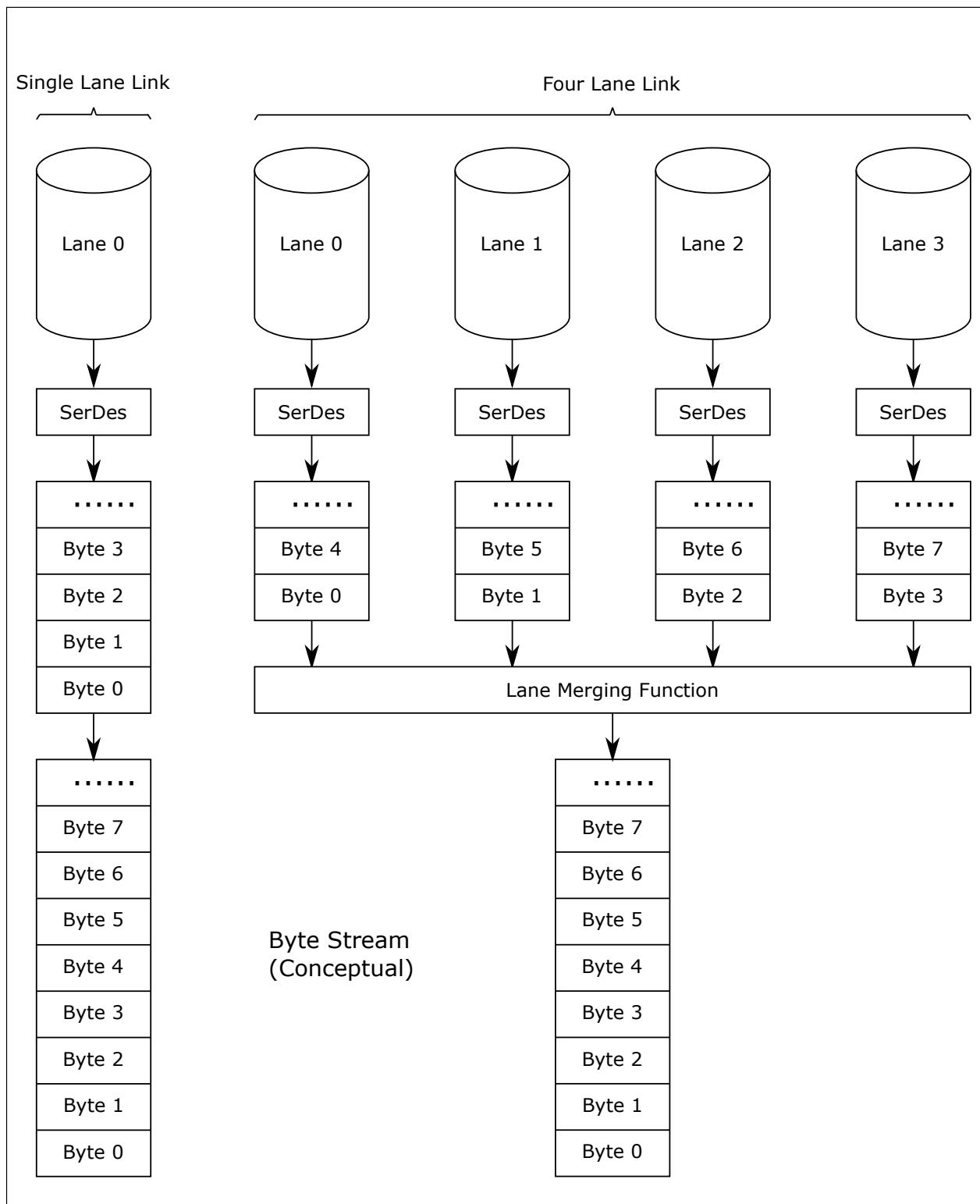


图 12.22: 接收端组合

通道分配器接受任意字节长度的 HS 传输，缓冲 N 个字节，其中 N 是接口中实现的通道数，并在 N 个通道上并行发送 N 个字节组。在发送数据之前，所有通道并行执行 SoT 序列，以向其对应的接收单元指示数据包的第一个字节开始。SoT 之后，通道通过循环过

程从第一个数据包并行发送  $N$  字节组，对于四通道系统，数据包的字节 0 进入通道 0，字节 1 进入通道 1，字节 2 进入通道 2，字节 3 进入通道 3，字节 4 进入通道 0，依此类推。由于 HS 传输由任意数量的字节组成，这些字节可能不是通道的整数倍，因此某些通道可能会在其他通道之前先发完数据。尽管所有通道均以并行 SoT 同时启动，但每条通道独立运行，可在其他通道之前完成 HS 传输，提前一个字节发送 EoT。如下图所示：

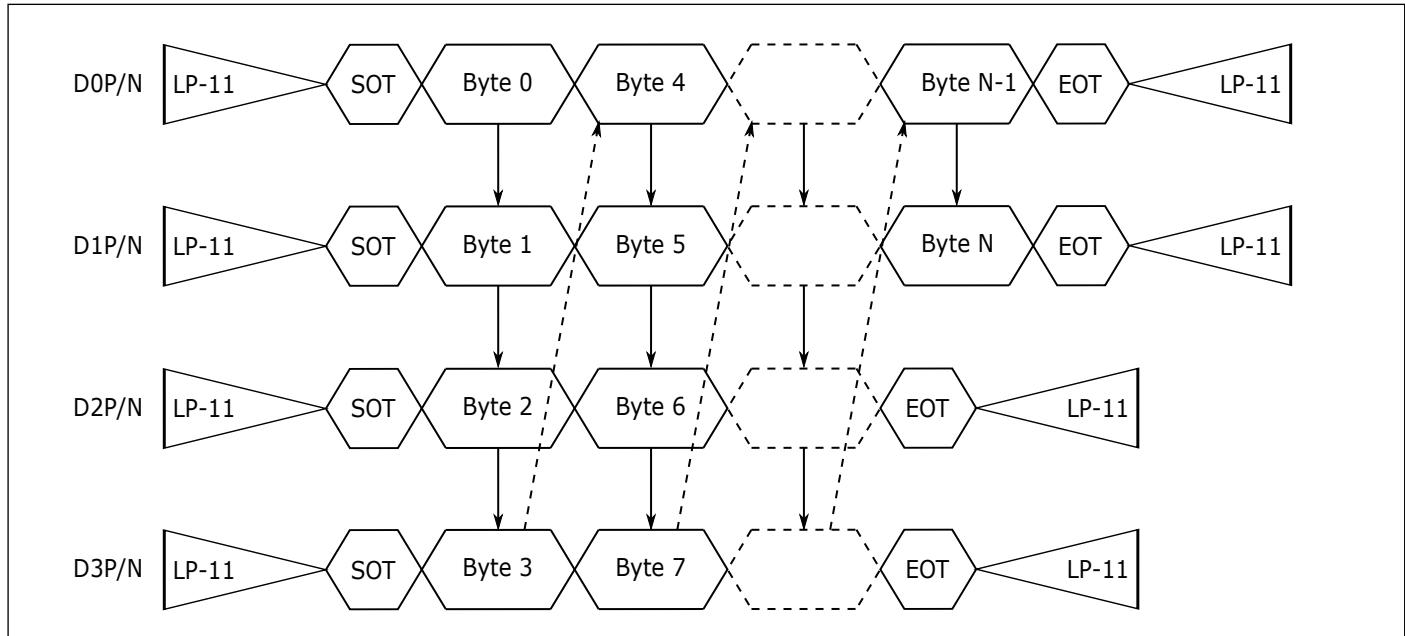


图 12.23: 非通道整数倍传输

## 12.3.6 协议层

DSI 是一个基于数据包传送的通信协议，MCU 和显示模块之间传送的命令和数据基本上都是以数据包格式进行。DSI 所定义的数据包有两种：短数据包 (SPa) 和长数据包 (LPa)。数据包的类型 (SPa 或 LPa) 可以从它们的包头 (PH) 中识别出来。

### 12.3.6.1 短数据包

短数据包由一个 8-bit 数据标识 (DI)、两个字节命令或数据和一个 8-bit ECC 组成。包括 ECC 在内，短数据包的长度为 4 字节。短数据包主要用于大多数命令模式命令和相关参数，另外还用来传递诸如 H 同步和 V 同步边缘之类的事件。因为长度较短，所以可以准确地将定时信息传送到外围逻辑。短数据包结构如下图所示：

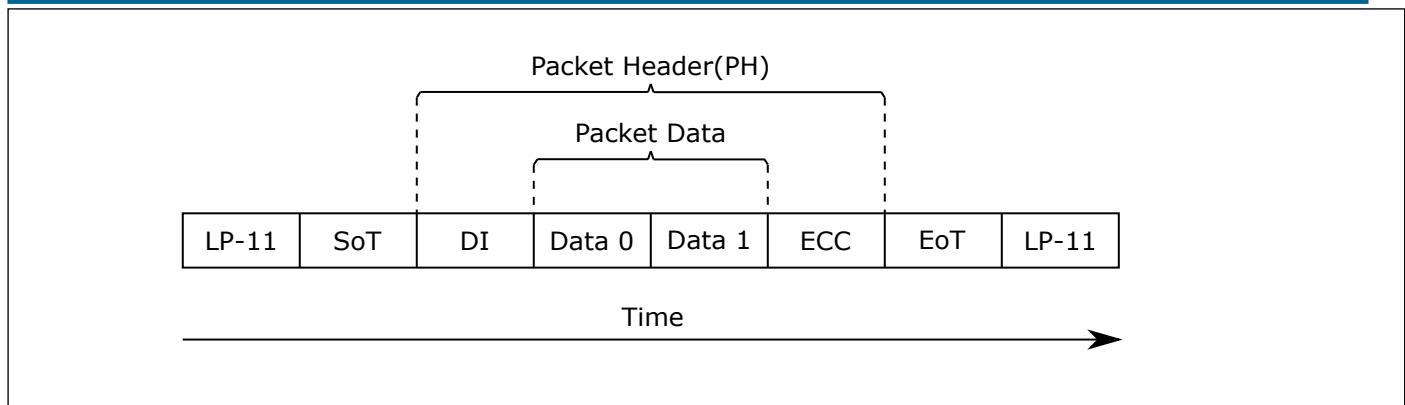


图 12.24: 短数据包结构图

其中：

- LP-11: 低功率-停止状态
- SoT: 传输开始
- DI: 8-bit, 数据标识
- Data 0: 8-bit, 包数据 0
- Data 1: 8-bit, 包数据 1
- ECC: 8-bit, 纠错码
- EoT: 传输结束

### 12.3.6.2 长数据包

长数据包由 32-bit 数据包头 (PH)、具有可变字节数的数据有效负载和 16-bit 数据包尾 (PF) 组成。其中包头可分为 8-bit 数据标识、16-bit 字计数 (WC) 和 8-bit ECC。由于记录有效负载长度的 WC 为两个字节，取值范围为 0 到 65535，因此长数据包的长度范围是 6 到 65541 个字节。长数据包主要用于传输大量图像数据或部分控制命令。长数据包结构如下图所示：

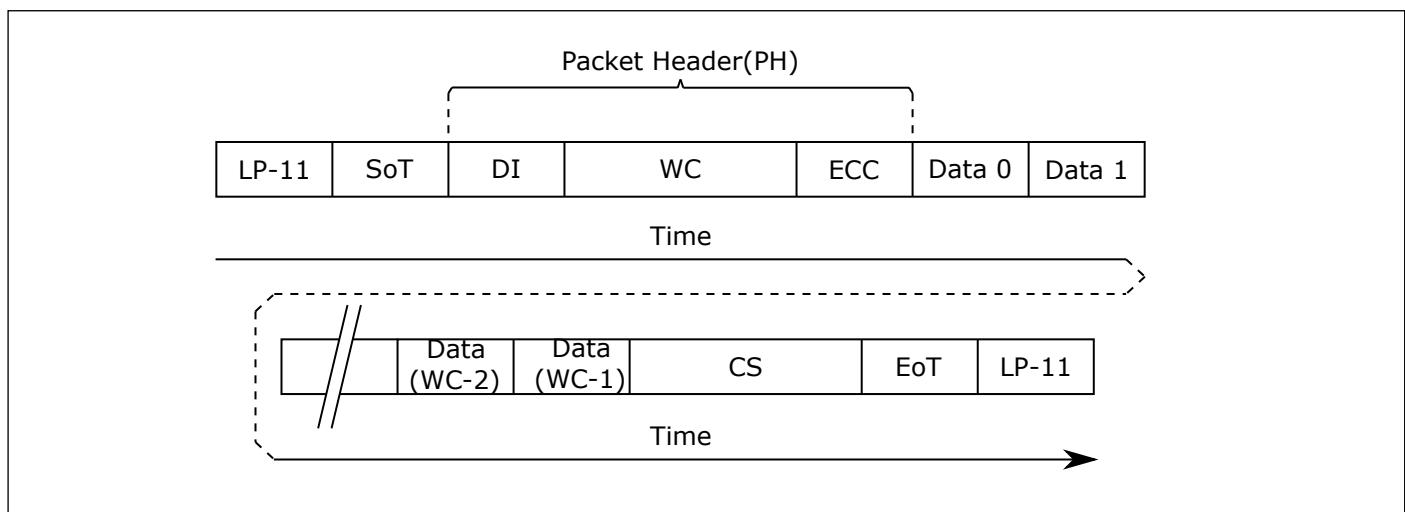


图 12.25: 长数据包结构图

其中：

- LP-11: 低功率-停止状态
- SoT: 传输开始
- DI: 8-bit, 数据标识
- WC: 16-bit, 字计数
- ECC: 8-bit, 纠错码
- Data 0, Data 1...: 数据包数据 (0~65535 字节)
- CS: 16-bit, 校验和
- EoT: 传输结束

### 12.3.6.3 每次传输多个数据包

在上面短数据包和长数据包结构图中，传输只包含了一个分组。对于多个数据包，如果每次传输都只发送一个数据包，LPS 和高速模式之间频繁切换的开销将严重限制带宽。DSI 协议允许将多个数据包连接起来发送，数据包之间不需要 SoT、EoT 和 LP-11，如下图所示：

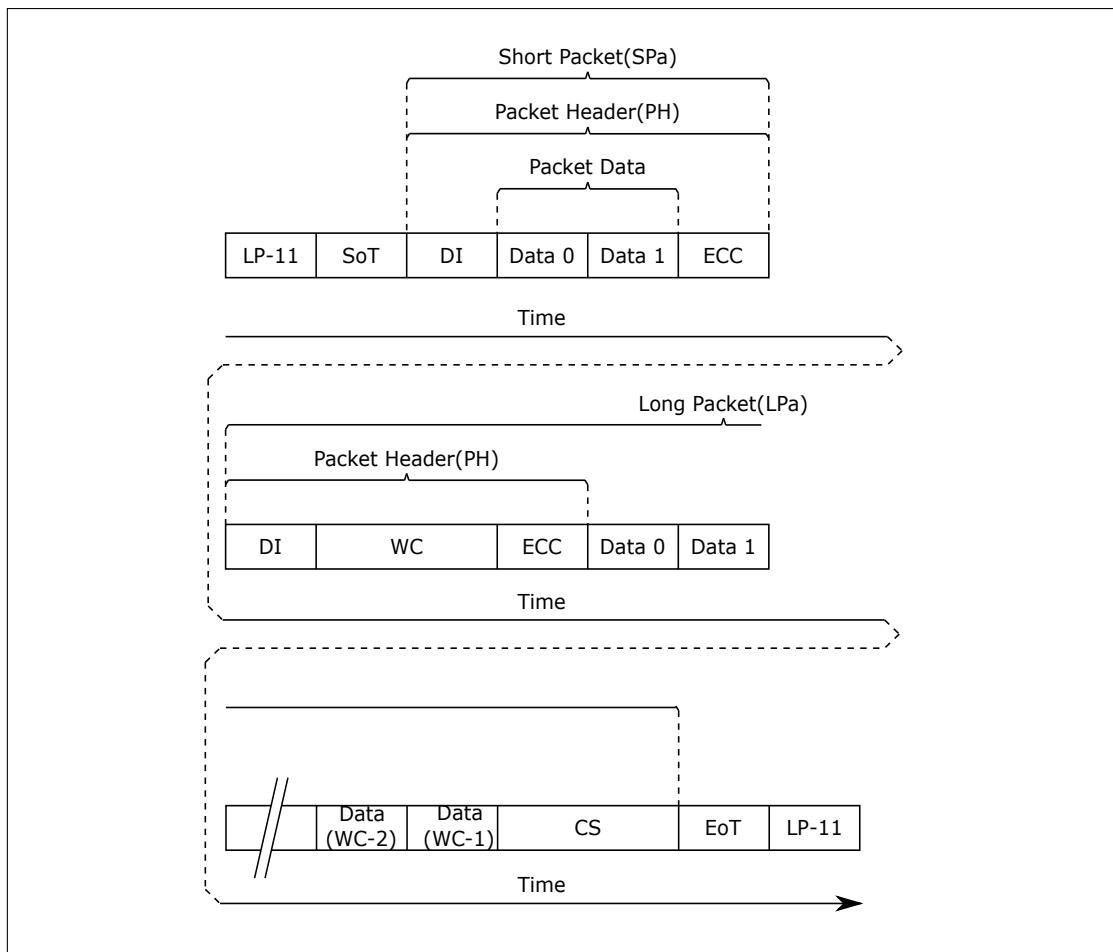


图 12.26: 一次传输多个数据包

### 12.3.6.4 数据包字节的位顺序

在数据包中使用的字节的位顺序是首先发送字节的最低有效位 (LSB)，最后发送最高有效位 (MSB)，下图是一个例子：

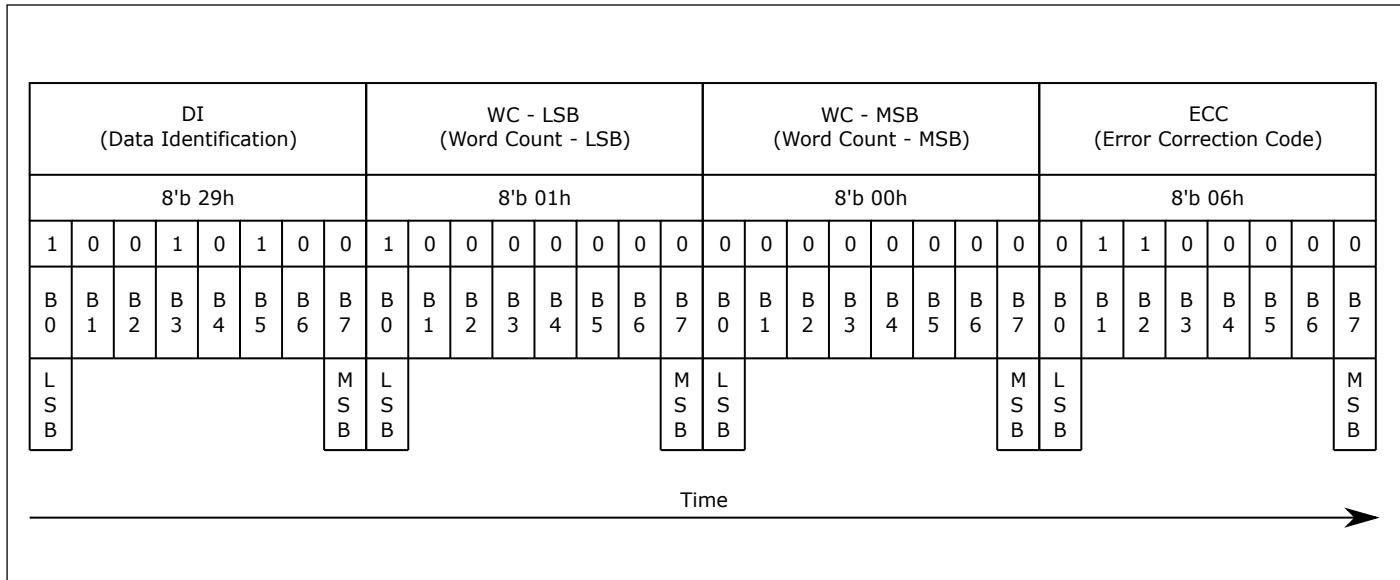


图 12.27: 数据包字节的位顺序

### 12.3.6.5 数据包的字节顺序

在数据包中使用的多字节信息的字节顺序是首先发送信息的最低有效 (LS) 字节，最后发送最高有效 (MS) 字节。例如字计数 (WC) 由 2 个字节组成，首先发送 LS 字节，然后发送 MS 字节，如下图所示：

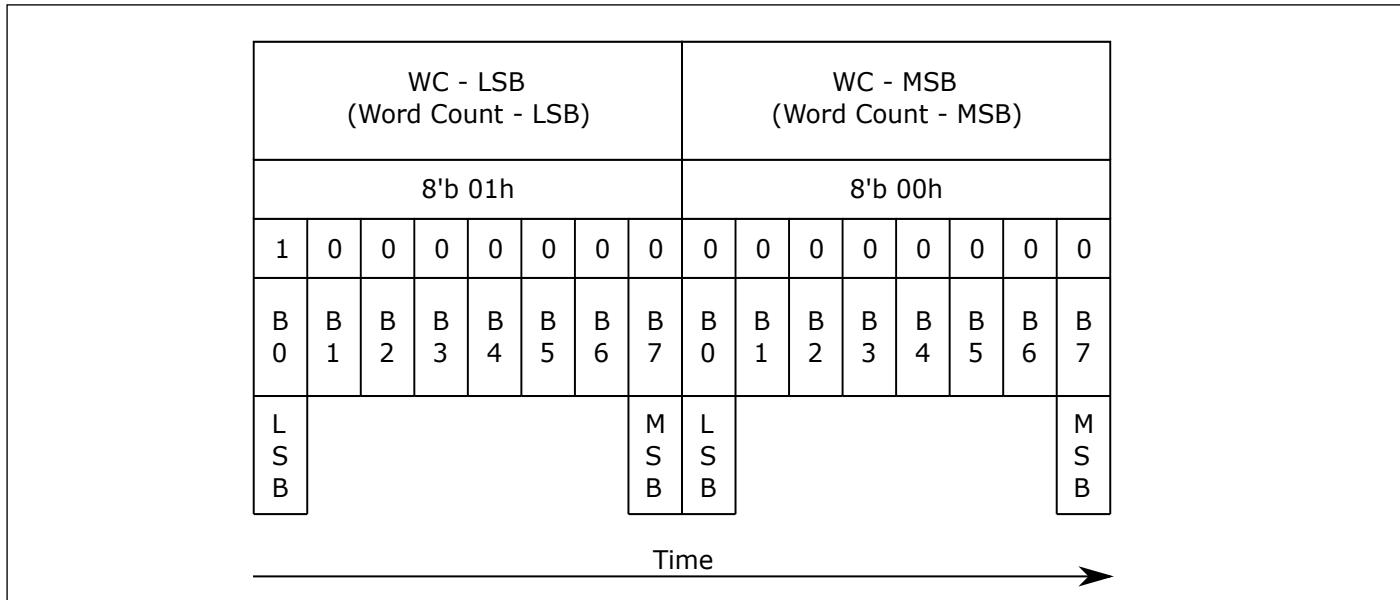


图 12.28: 数据包的字节顺序

### 12.3.6.6 数据包头 (PH)

数据包头始终由 4 个字节组成，对于短数据包和长数据包，这 4 个字节的内容有所不同。短数据包包括：1 个字节的数据标识 (DI)、2 个字节的数据包数据 (PD0、PD1) 和 1 个字节的纠错码 (ECC)。长数据包包括：1 个字节的数据标识 (DI)、2 个字节的字计数 (WC) 和 1 个字节的纠错码 (ECC)。

### 12.3.6.7 数据标识 (DI)

数据标识是数据包头 (PH) 的一部分，它由两个部分组成：

- 虚拟通道 (VC)，2-bit，DI[7:6]
- 数据类型 (DT)，6-bit，DI[5:0]

### 12.3.6.8 虚拟通道 (VC)

虚拟通道是数据标识 (DI[7:6]) 结构的一部分，用于确定 MCU 数据包发往的位置。虚拟通道可以为 4 个不同的显示模块分配 4 个不同的通道，各个显示模块将使用与 MCU 用于向其发送数据包的虚拟通道相同的虚拟通道。

### 12.3.6.9 数据类型 (DT)

数据类型是数据标识 (DI[5:0]) 结构的一部分，用于定义数据包是长数据包类型还是短数据包类型以及数据包格式。其中从 MCU 到显示模块的数据类型的定义如下表所示：

From the MCU to the Display Module								
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Hex	Description	Short/Long Packet
0	0	0	0	0	1	01	Sync Event, V Sync Start	SPa (Short Packet)
0	1	0	0	0	1	11	Sync Event, V Sync End	SPa (Short Packet)
1	0	0	0	0	1	21	Sync Event, H Sync Start	SPa (Short Packet)
1	1	0	0	0	1	31	Sync Event, H Sync End	SPa (Short Packet)
0	0	1	0	0	0	08	End of Transmission Packet (EoTP), Note 1	SPa (Short Packet)
0	0	0	0	1	0	02	Color Mode Off Command	SPa (Short Packet)
0	1	0	0	1	0	12	Color Mode On Command	SPa (Short Packet)
1	0	0	0	1	0	22	Shut Down Peripheral Command	SPa (Short Packet)
1	1	0	0	1	0	32	Turn On Peripheral Command	SPa (Short Packet)
0	0	0	0	1	1	03	Generic Short Write, No Parameter	SPa (Short Packet)
0	1	0	0	1	1	13	Generic Short Write, 1 Parameter	SPa (Short Packet)
1	0	0	0	1	1	23	Generic Short Write, 2 Parameters	SPa (Short Packet)
0	0	0	1	0	0	04	Generic Short Read, No Parameter	SPa (Short Packet)
0	1	0	1	0	0	14	Generic Short Read, 1 Parameter	SPa (Short Packet)
1	0	0	1	0	0	24	Generic Short Read, 2 Parameters	SPa (Short Packet)
0	0	0	1	0	1	05	DCS Write, No Parameter	SPa (Short Packet)
0	1	0	1	0	1	15	DCS Write, 1 Parameter	SPa (Short Packet)
0	0	0	1	1	0	06	DCS Read, No Parameter	SPa (Short Packet)
1	1	0	1	1	1	37	Set Maximum Return Packet Size	SPa (Short Packet)
0	0	1	0	0	1	09	Null Packet, No Data, Note 2	LPa (Long Packet)
0	1	1	0	0	1	19	Blanking Packet, No Data	LPa (Long Packet)
1	0	1	0	0	1	29	Generic Long Write	LPa (Long Packet)
1	1	1	0	0	1	39	DCS Write Long	LPa (Long Packet)
0	0	1	1	1	0	0E	Packed Pixel Stream, 16-bit RGB, 5-6-5 Format	LPa (Long Packet)
0	1	1	1	1	0	1E	Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	LPa (Long Packet)
1	0	1	1	1	0	2E	Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	LPa (Long Packet)
1	1	1	1	1	1	3E	Packed Pixel Stream, 24-bit RGB, 8-8-8 Format	LPa (Long Packet)
x	x	0	0	0	0	x0	Do Not Use	
x	x	1	1	1	1	xF	All Unspecified Codes Are Reserved	

**Note:**

1. This item can be used when the MCU wants to confirm the end of the transmission in the high-speed data transmission mode;
2. This item can be used when the data channel remains in high-speed data transfer mode.

图 12.29: 从 MCU 到显示模块的数据类型

从显示模块到 MCU 的数据类型的定义如下表所示:

From the Display Module to the MCU								
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Hex	Description	Short/Long Packet
0	0	0	0	1	0	02	Acknowledge with Error Report	SPa (Short Packet)
0	0	1	0	0	0	08	End of Transmission Packet (EoTP)	SPa (Short Packet)
0	1	0	0	0	1	11	Generic Short Read Response, 1 Byte Returned	SPa (Short Packet)
0	1	0	0	1	0	12	Generic Short Read Response, 2 Bytes Returned	SPa (Short Packet)
0	1	1	0	1	0	1A	Generic Long Read Response	LPa (Long Packet)
0	1	1	1	0	0	1C	DCS Read Long Response	LPa (Long Packet)
1	0	0	0	0	1	21	DCS Read Short Response, 1 Byte Returned	SPa (Short Packet)
1	0	0	0	1	0	22	DCS Read Short Response, 2 Bytes Returned	SPa (Short Packet)

图 12.30: 从显示模块到 MCU 的数据类型

### 12.3.6.10 包数据 (PD)

对于短数据包，包数据是数据包头的一部分，位于数据标识之后，固定 2 个字节，即数据 0 和数据 1。包数据的发送顺序是先发送数据 0，再发送数据 1。如果信息长度为 1 个字节，则数据 1 的值为 0。对于长数据包，包数据位于数据包头之后，字节长度由字计数 (WC) 确定。

### 12.3.6.11 字计数 (WC)

字计数是长数据包包头的一部分，位于数据标识之后，用于确定长数据包的包数据 (PD) 部分的总字节数。字计数固定为 2 个字节，发送顺序是先发送最低有效字节，再发送最高有效字节。

### 12.3.6.12 纠错码 (ECC)

纠错码是数据包头的一部分，可以识别一个或多个错误，并且只能纠正一个错误。ECC 保护以下字段：- 短数据包：DI(8-bit)、PD(16-bit) 和 ECC(8-bit) - 长数据包：DI(8-bit)、WC(16-bit) 和 ECC(8-bit) 以短数据包为例，ECC 各位的计算方法如下图所示（“^”表示异或运算）：

DI (Data Identification)								Data 0 (Packet Data)								Data 1 (Packet Data)								ECC (Error Correction Code)									
8'b 05h								8'b 10h								8'b 00h								8'b 2Ch									
1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	
D 0	D 1	D 2	D 4	D 5	D 7	D 10	D 11	D 13	D 16	D 20	D 21	D 22	D 23	P 0																			
D 0	D 1	D 3	D 4	D 6	D 8	D 10	D 12	D 14	D 17	D 20	D 21	D 22	D 23	P 1																			
D 0	D 2	D 3	D 5	D 6	D 9	D 11	D 12	D 15	D 18	D 20	D 21	D 22	D 23	P 2																			
D 1	D 2	D 3	D 7	D 8	D 9	D 13	D 14	D 15	D 19	D 20	D 21	D 23	D 23	P 3																			
			D 4	D 5	D 6	D 7	D 8	D 9	D 16	D 17	D 18	D 19	D 20	D 22	D 23	P 4																	
						D 10	D 11	D 12	D 13	D 14	D 15	D 16	D 17	D 18	D 19	D 21	D 22	D 23	D 23	D 23	D 23	D 23	D 23	D 23	P 5								
B 0	B 1	B 2	B 3	B 4	B 5	B 6	B 7	B 0	B 1	B 2	B 3	B 4	B 5	B 6	B 7	B 0	B 1	B 2	B 3	B 4	B 5	B 6	B 7	B 0	B 1	B 2	B 3	B 4	B 5	B 6	B 7		
L S B								M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B	L S B	M S B			

Time →

图 12.31: ECC 计算示例

公式:

- $ECC[7] = 0$
- $ECC[6] = 0$
- $ECC[5] = D10 \wedge D11 \wedge D12 \wedge D13 \wedge D14 \wedge D15 \wedge D16 \wedge D17 \wedge D18 \wedge D19 \wedge D21 \wedge D22 \wedge D23$
- $ECC[4] = D4 \wedge D5 \wedge D6 \wedge D7 \wedge D8 \wedge D9 \wedge D16 \wedge D17 \wedge D18 \wedge D19 \wedge D20 \wedge D22 \wedge D23$
- $ECC[3] = D1 \wedge D2 \wedge D3 \wedge D7 \wedge D8 \wedge D9 \wedge D13 \wedge D14 \wedge D15 \wedge D19 \wedge D20 \wedge D21 \wedge D23$
- $ECC[2] = D0 \wedge D2 \wedge D3 \wedge D5 \wedge D6 \wedge D9 \wedge D11 \wedge D12 \wedge D15 \wedge D18 \wedge D20 \wedge D21 \wedge D22$
- $ECC[1] = D0 \wedge D1 \wedge D3 \wedge D4 \wedge D6 \wedge D8 \wedge D10 \wedge D12 \wedge D14 \wedge D17 \wedge D20 \wedge D21 \wedge D22 \wedge D23$
- $ECC[0] = D0 \wedge D1 \wedge D2 \wedge D4 \wedge D5 \wedge D7 \wedge D10 \wedge D11 \wedge D13 \wedge D16 \wedge D20 \wedge D21 \wedge D22 \wedge D23$

### 12.3.6.13 数据包尾 (PF)

数据包尾是长数据包的一部分，位于包数据 (PD) 之后。包尾是根据长数据包的包数据计算的校验和，该校验和是使用由多项式  $X^{16}+X^{12}+X^5+X^0$  生成的 16 位循环冗余校验 (CRC) 值，如下图所示：

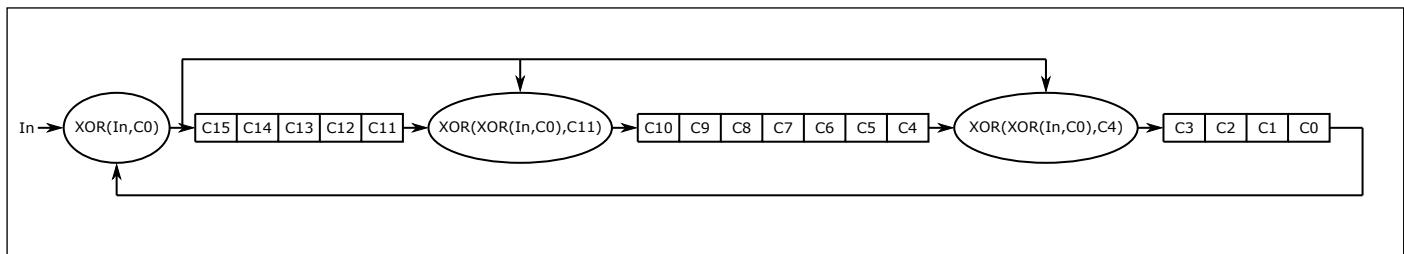


图 12.32: CRC 计算方法

16 位循环冗余校验 (CRC) 发生器在计算前的初始值为 0xFFFF，包数据最低有效位 (LSB) 是输入 CRC 的第一位，下图是以数据 0x01 为例，进行 CRC 计算的步骤说明：

Step	In	XOR(In, C0)	C15	C14	C13	C12	C11	XOR(XOR(In, C0), C11(Step-1))	C10	C9	C8	C7	C6	C5	C4	XOR(XOR(In, C0), C4(Step-1))	C3	C2	C1	C0	C0	
0	X	X	1	1	1	1	1	X	1	1	1	1	1	1	1	X	1	1	1	1	X	
1 1(LSB)	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2	0	1	1	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	
3	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	
4	0	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0	0	0	0	1	1	
5	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	
6	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	
7	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	
8 0(MSB)	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0
	1 byte	CRC result	0	0	0	0	1	1		1	1	0	0	0	0	0	1	1	1	0		LSB
								MSB														

图 12.33: CRC 计算示例

### 12.3.7 应用层

该层描述数据流中包含的数据的更高级编码和解释。取决于显示子系统架构，它可以由具有规定格式的像素或编码比特流，或者由显示模块内的显示控制器解释的命令组成。DSI 规范描述了像素值、比特流、命令和命令参数到数据包组件中字节的映射。(具体内容见显示命令集 (DCS) 规范，DCS 规范不在本文档介绍范围内)

## 12.3.8 命令模式

命令模式是指以发送命令和数据的形式实现 MCU 与包含显示控制器的外围设备(如显示模块)的交互。显示控制器可以包括本地寄存器和压缩或未压缩帧缓冲器，使用命令模式的系统对寄存器和帧缓冲存储器进行写入和读取。MCU 通过向显示控制器发送命令、参数和数据，间接控制外围设备的活动。MCU 还可以读取显示模块状态信息或帧存储器的内容。命令模式需要双向接口。

## 12.3.9 视频模式

视频模式是指以实时像素流的形式从 MCU 传输到外围设备的操作。显示模块依靠 MCU 以足够的带宽提供图像数据，以避免显示图像中的闪烁或其他可见伪影。视频信息只能使用高速模式传输。视频模式又可以分为三种模式：具有同步脉冲的非突发模式 (Non-Burst Mode with SYNC Pulses)、具有同步事件的非突发模式 (Non-Burst Mode with SYNC Events) 和突发模式 (Burst Mode)。这三种模式有着不同的数据包序列，外围设备的定时要求决定了哪种模式是合适的。

### 12.3.9.1 具有同步脉冲的非突发模式 (Non-Burst Mode with SYNC Pulses)

该模式使外围设备能够准确重建原始视频定时，包括同步脉冲宽度。使用这种模式，目标是通过 DSI 串行链路准确地传送 DPI 类型的定时，这包括匹配的 DPI 像素传输速率和同步脉冲等定时事件的宽度。因此，使用发送同步脉冲的开始和结束的数据包来定义同步周期。此模式的示例如下图所示，其中 HSA、HBP 和 HFP 的周期由 LP 模式中的定时间隔来填充，DSI 会在 HSA、HBP 和 HFP 开始时从 HS 切换到 LP 模式，期间保持 LP-11 状态，在结束时切换回 HS 模式：

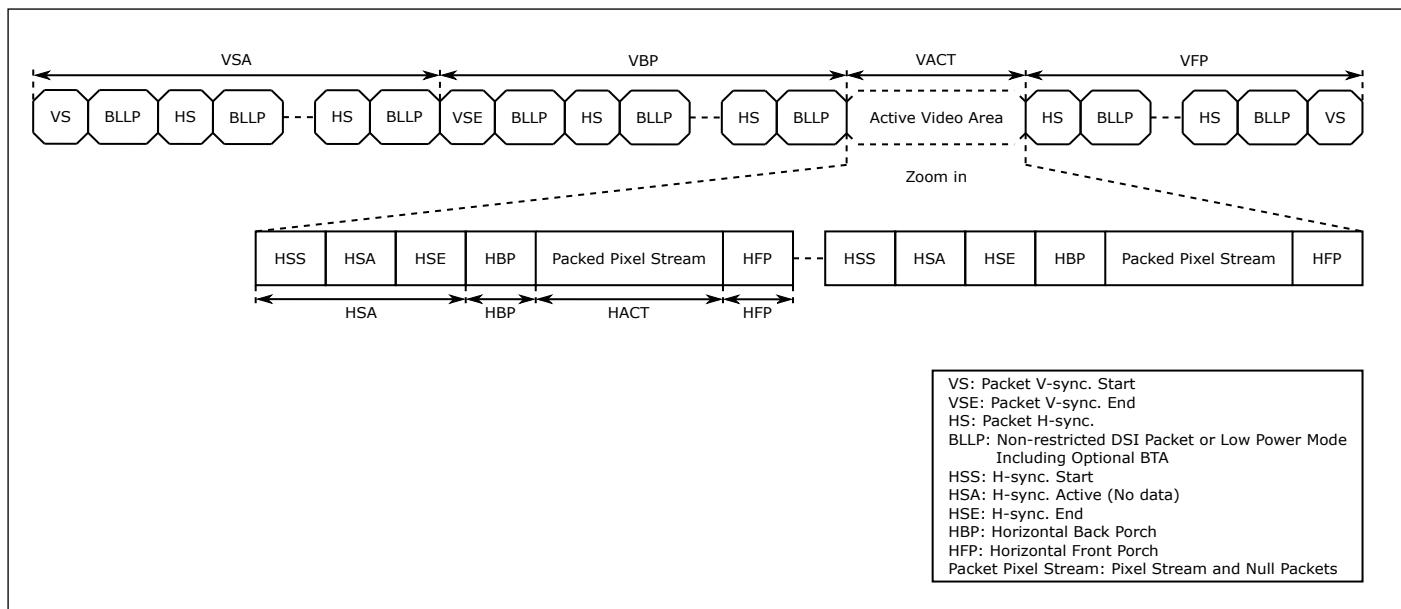


图 12.34: 具有同步脉冲的非突发模式时序图

### 12.3.9.2 具有同步事件的非突发模式 (Non-Burst Mode with SYNC Events)

该模式与具有同步脉冲的非突发模式类似，但不需要精确重建同步脉冲宽度，因此将同步脉冲替换为单个同步事件，是对上述具有同步事件的非突发模式的简化。这种模式仅传输每个同步脉冲的开始，外围设备可根据需要从接收到的每个同步事件包中重新生成同步脉冲。像素的传输速率与在相应的并行显示接口（如 DPI-2）中的传输速率相同。此模式的示例如下图所示，其中 HSA、HBP 和 HFP 的周期由 LP 模式中的定时间隔来填充，DSI 会在 HSA、HBP 和 HFP 开始时从 HS 切换到 LP 模式，期间保持 LP-11 状态，在结束时切换回 HS 模式：

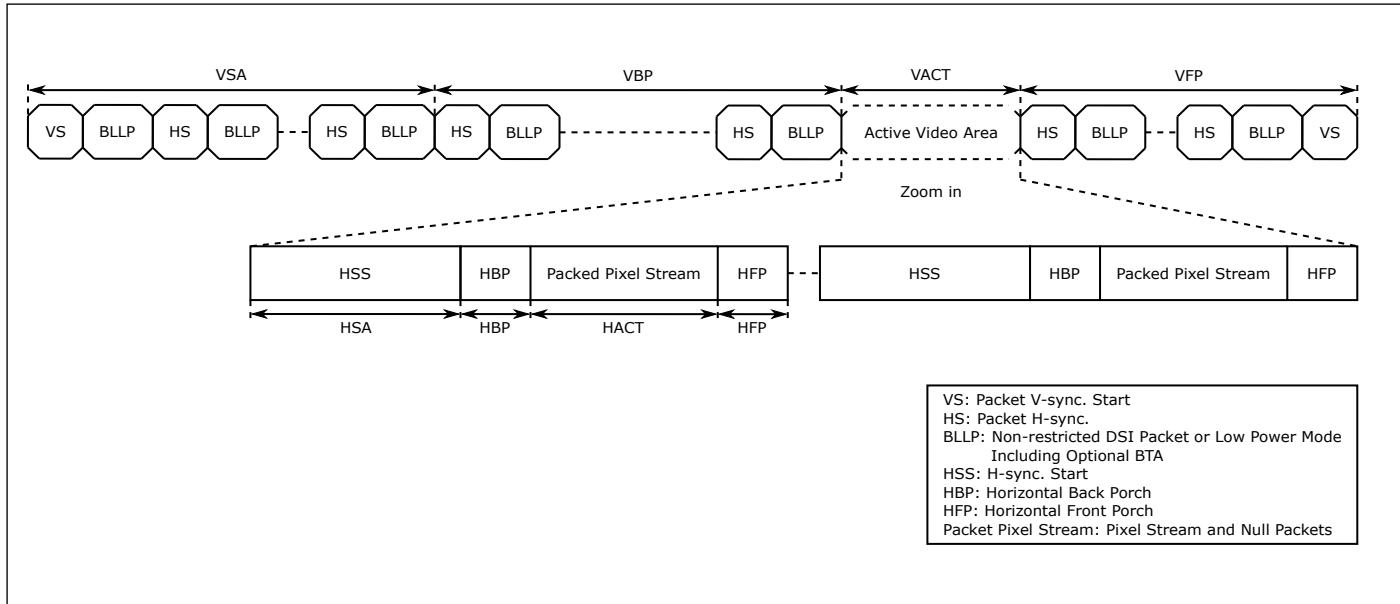


图 12.35: 具有同步事件的非突发模式时序图

### 12.3.9.3 突发模式 (Burst Mode)

该模式下的像素数据包经过了时间压缩，从而可以在较短时间内完成传输。这种方式降低了总体 DSI 功耗，并为链路上任何方向的其他数据传输提供更大的时间块。在 HS 像素数据传输完成之后，总线会进入 LP 模式。在此期间，总线可以保持空闲即 LP-11 状态，或者 LP 传输可以在任一方向上进行。如果外围设备控制向 MCU 发送数据的总线，其传输时间应受到限制，以确保数据不会从其内部缓冲存储器流向显示设备。此模式的示例如下图所示，其中 HSA、HBP 和 HFP 的周期由 LP 模式中的定时间隔来填充，DSI 会在 HSA、HBP 和 HFP 开始时从 HS 切换到 LP 模式，期间保持 LP-11 状态，在结束时切换回 HS 模式：

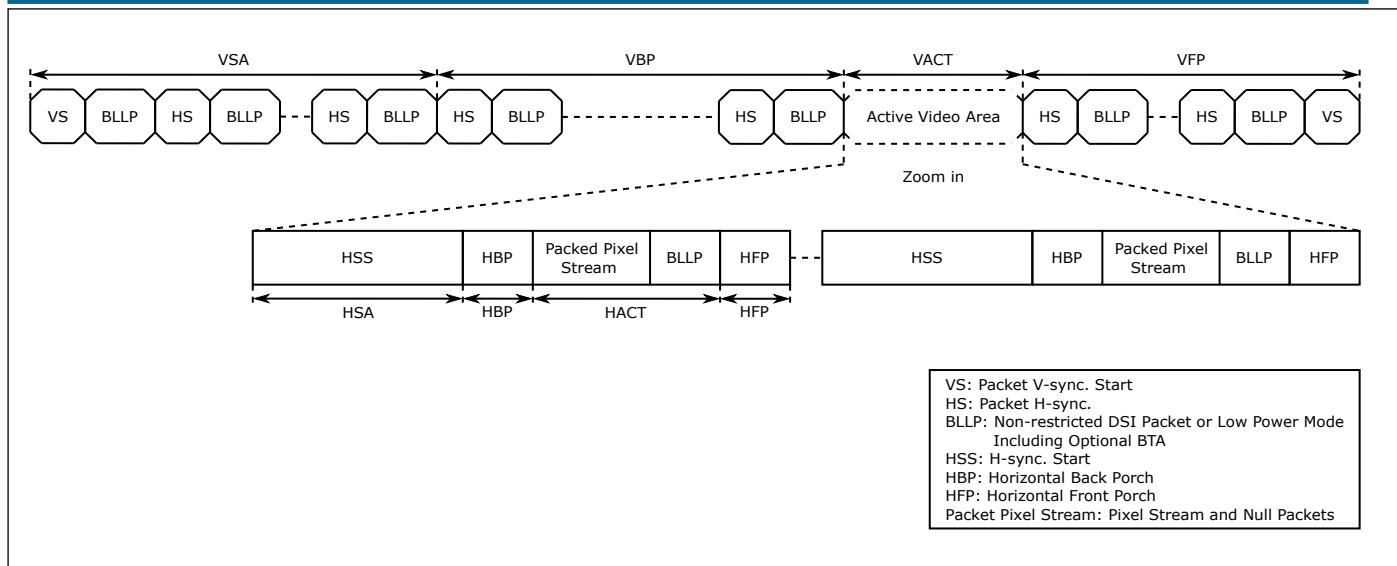


图 12.36: 突发模式时序图

### 12.3.10 Line Buffer

因为前端模块向 DSI 模块输入数据的速度和 DSI 输出数据的速度不匹配，所以需要对数据进行缓冲。DSI 内部有一个最多可以容纳 1280\*3 个字节的 Line Buffer，用于视频模式下对输入 DSI 模块的数据进行缓存。用户需要通过 `<CR_HSTX_OUT_TH>` 设置一个阈值，对于视频模式下的每行 (H-Sync) 图像数据，DSI 模块在收到其前端模块输入的数据之后不会将数据立即发送出去，而是先缓存在 Line Buffer 中。当 Line Buffer 中缓存的像素点达到设定的阈值时，DSI 开始向显示模块发送数据。这样当 DSI 模块的输入速度大于输出速度时，输入部分来不及发送的数据会先放在 Line Buffer 中；而当 DSI 模块的输入速度小于输出速度时，Line Buffer 会先缓存一部分数据，达到阈值后再开始发送，以保证 DSI 发送整行像素的时间与剩余像素输入的时间相匹配。阈值的具体计算公式如下：

$$\text{Threshold} = \text{ceil}(\text{Width} * (1 - \text{Fdp} * \text{BPP} / \text{Fhs} / \text{LN}))$$

其中：

- `ceil()` 为向上取整；
- `Width` 为图像一行的像素点数；
- `Fdp` 为 DSI 上一级模块 (即 DSI 的数据输入模块 `dp_dvp_tsrc`) 的工作时钟；
- `BPP`(Byte Per Pixel) 为像素格式的字节数，RGB888/RGB666 为 3，RGB565/YUV422\_8 为 2；
- `LN`(Lane Number) 为 Data Lane 的数量；
- `Threshold` 为所求的阈值，阈值的最小值为 6，当计算值小于 6 时应设置为 6。

## 12.3.11 显示数据格式

显示数据格式支持 YUV422(8-bit)、RGB565、RGB666(loosely packed) 和 RGB888 四种，在 <CR\_DT> 中进行配置。

### 12.3.11.1 YUV422(8-bit)

YUV422(8-bit) 格式中一个像素的 Y、U、V 分量各为 8-bit，其中两个像素共用一组 UV。YUV422(8-bit) 格式的长数据包由一字节 DI、两字节非零 WC、一字节 ECC、长度为 WC 字节的有效负载和两字节 CRC 组成。使用这种格式，像素边界与某些字节边界对齐，WC 中的值应为任何可被 4 整除的非零值。YUV422(8-bit) 格式长数据包结构如下图所示：

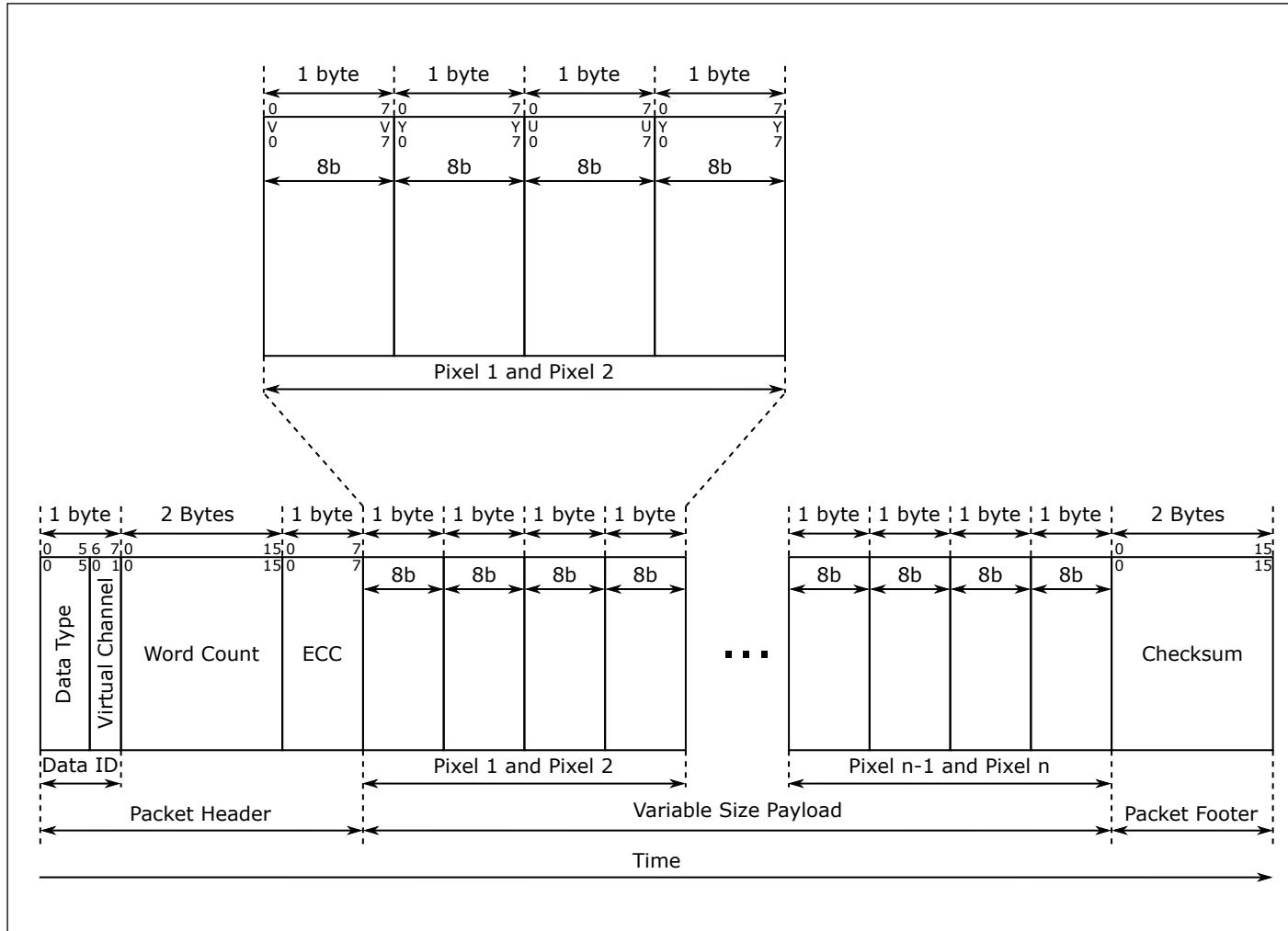


图 12.37: YUV422 像素格式

### 12.3.11.2 RGB565

RGB565 格式中一个像素的 R、G、B 分量分别为 5-bit、6-bit、5-bit。RGB565 格式的长数据包由一字节 DI、两字节 WC、一字节 ECC、长度为 WC 字节的有效负载和两字节 CRC 组成。使用这种格式，像素边界每两个字节与字节边界对齐一次，WC 中的值应为 2 的倍数。RGB565 格式长数据包结构如下图所示：

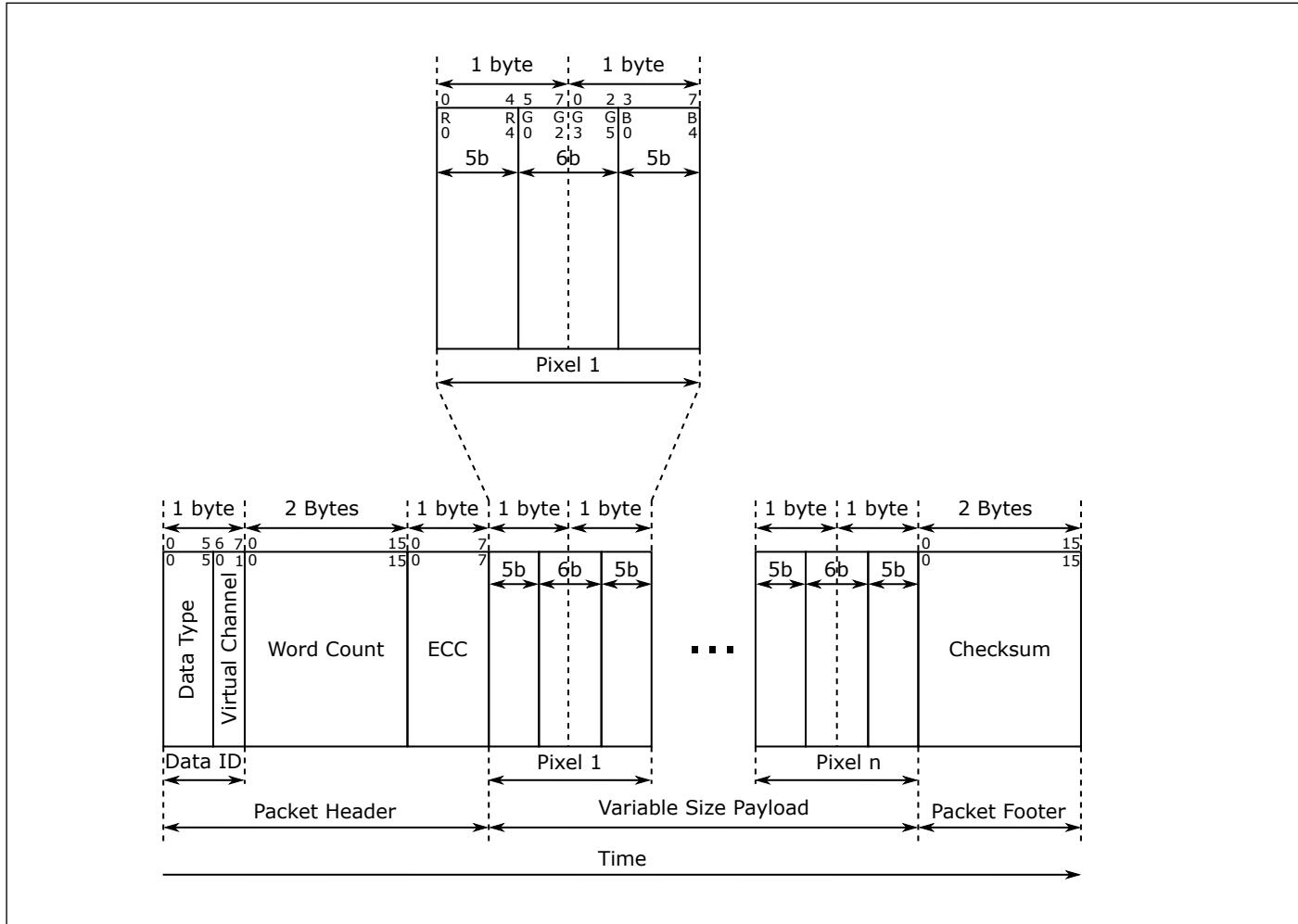


图 12.38: RGB565 像素格式

### 12.3.11.3 RGB666(loosely packed)

RGB666(loosely packed) 格式中一个像素的 R、G、B 分量各为 6-bit，但会移动到字节的高位，以便有效像素位占据每个字节的位 [7:2]，而位 [1:0] 会被忽略。因此，每个像素在通道上传输时需要三个字节。RGB666 格式的长数据包由一字节 DI、两字节 WC、一字节 ECC、长度为 WC 字节的有效负载和两字节 CRC 组成。使用这种格式，像素边界每三个字节与字节边界对齐一次，WC 中的值应为 3 的倍数。RGB666(loosely packed) 格式长数据包结构如下图所示：

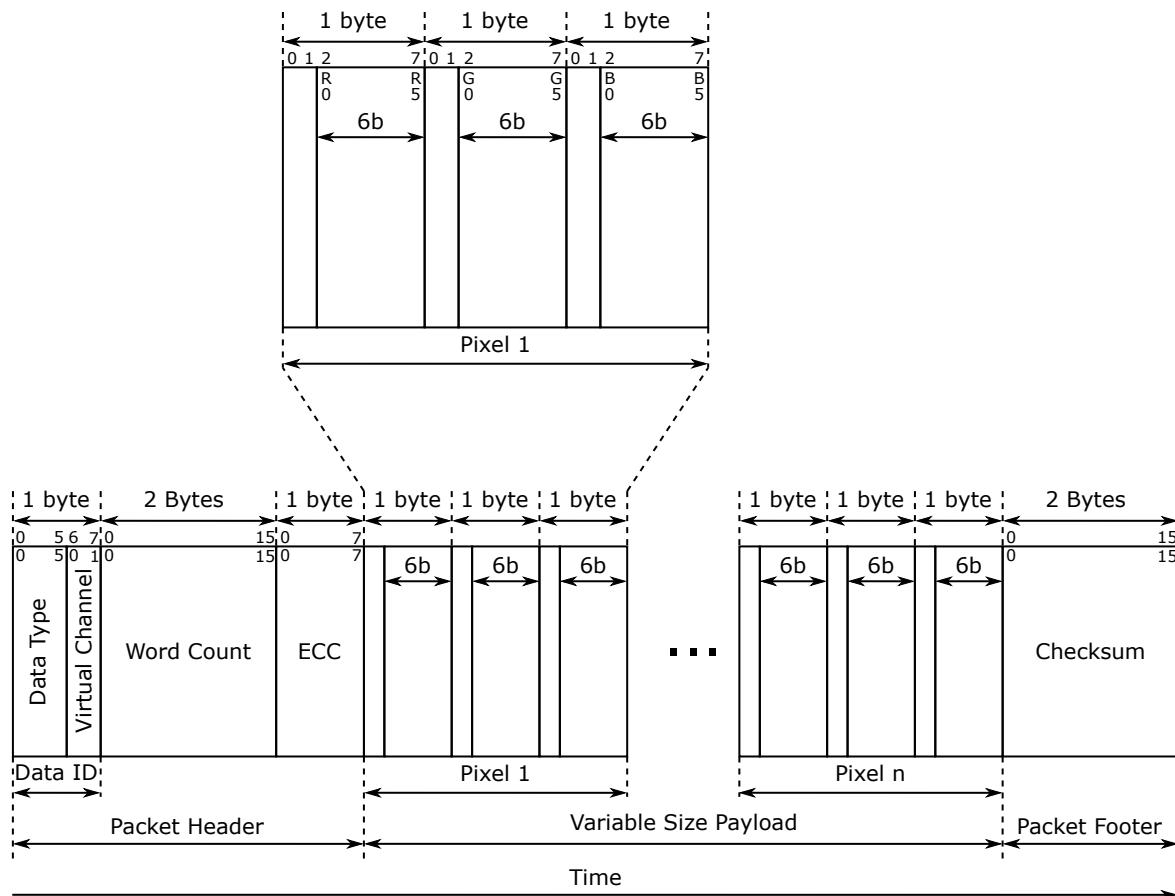


图 12.39: RGB666 像素格式

#### 12.3.11.4 RGB888

RGB888 格式中一个像素的 R、G、B 分量各为 8-bit。RGB888 格式的长数据包由一字节 DI、两字节 WC、一字节 ECC、长度为 WC 字节的有效负载和两字节 CRC 组成。使用这种格式，像素边界每三个字节与字节边界对齐一次，WC 中的值应为 3 的倍数。RGB888 格式长数据包结构如下图所示：

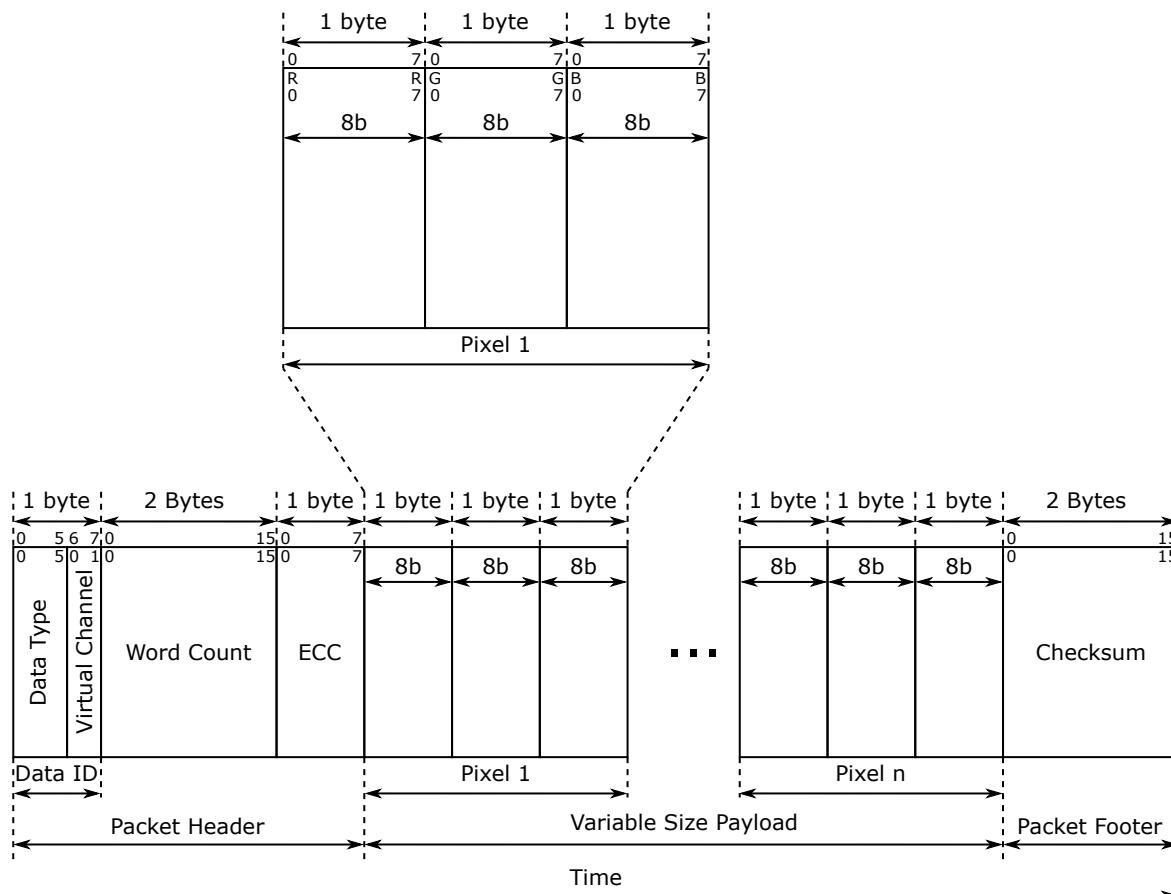


图 12.40: RGB888 像素格式

### 12.3.12 中断

DSI 模块拥有丰富的中断控制，包括以下几种：

- TX Escape 命令结束中断
- RX LPDT 结束中断
- RX ULPS 命令中断
- RX Trigger 0 命令中断
- RX Trigger 1 命令中断
- RX Trigger 2 命令中断
- RX Trigger 3 命令中断
- TX LPDT FIFO 请求中断
- RX LPDT FIFO 请求中断
- Buffer Overrun 错误中断
- Buffer Underrun 错误中断

- 像素过少错误中断
- 像素过多错误中断
- FIFO 溢出错误中断

TX Escape 命令结束中断会在任意 Escape 命令发送结束时触发；

RX LPDT 结束中断、RX ULPS 命令中断、RX Trigger 0 命令中断、RX Trigger 1 命令中断、RX Trigger 2 命令中断、RX Trigger 3 命令中断分别会在对应的命令发出去之后，收到对方回复的 ACK 之后触发；

当 DSI\_FIFO\_CONFIG\_1 中 TFICNT 大于 TFITH 时，产生 TX FIFO 请求中断，当条件不满足时该中断标志会自动清除；

当 DSI\_FIFO\_CONFIG\_1 中 RFICNT 大于 RFITH 时，产生 RX FIFO 请求中断，当条件不满足时该中断标志会自动清除；

Buffer Overrun 错误中断会在 Line Buffer Overrun，即进入 Line Buffer 的数据不能及时送出，Line Buffer 被塞满时产生；

Buffer Underrun 错误中断会在 Line Buffer Underrun，即 Line Buffer 中的数据被全部读出，新的数据不能及时填入时产生；

像素过少错误中断会在 H-Sync 期间发送的像素数量比 <CR\_HSTX\_PC> 配置值小时产生；

像素过多错误中断会在 H-Sync 期间发送的像素数量比 <CR\_HSTX\_PC> 配置值大时产生；

FIFO 溢出错误中断会在 TX 或者 RX 发生 Overflow 或者 Underflow 时产生。

### 12.3.13 DMA

DSI LPDT 模式下支持 DMA 传输模式，使用该模式需要通过寄存器 DSI\_FIFO\_CONFIG\_1 的位 <TFITH> 和 <RFITH> 分别设置 TX 和 RX FIFO 的阈值。当该模式启用后，如果 <TFICNT> 大于 <TFITH>，则会触发 DMA TX 请求，配置好 DMA 后，DMA 在收到该请求时，会按照设定从内存中将数据搬运到 TX FIFO。如果 <RFICNT> 大于 <RFITH>，则会触发 DMA RX 请求，配置好 DMA 后，DMA 在收到该请求时，会按照设定将 RX FIFO 的数据搬运到内存。

### 12.3.14 与 OSD 配合使用

DSI 的数据输入可以配置为先经过 OSD 处理，OSD 的功能介绍请参阅 OSD 模块。

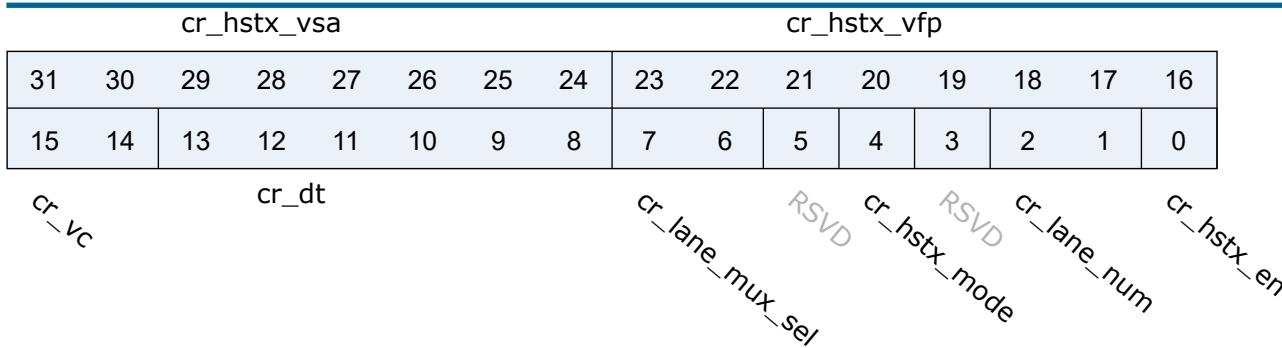
## 12.4 寄存器描述

名称	描述
dsi_config	
dsi_esc_config	
dsi_lpdt_tx_config	
dsi_hstx_config	
dsi_int_status	
dsi_int_mask	
dsi_int_clear	
dsi_int_enable	

名称	描述
dsi_fifo_config_0	
dsi_fifo_config_1	
dsi_fifo_wdata	
dsi_fifo_rdata	
dphy_config_0	
dphy_config_1	
dphy_config_2	
dphy_config_3	
dphy_config_4	
dphy_config_5	
dphy_config_6	
dphy_config_7	
dphy_config_8	
dphy_config_9	
dphy_config_10	
dphy_config_11	
dphy_config_12	
dphy_config_13	
dphy_config_14	
dphy_config_15	
dphy_config_16	
dummy_reg	

#### 12.4.1 dsi\_config

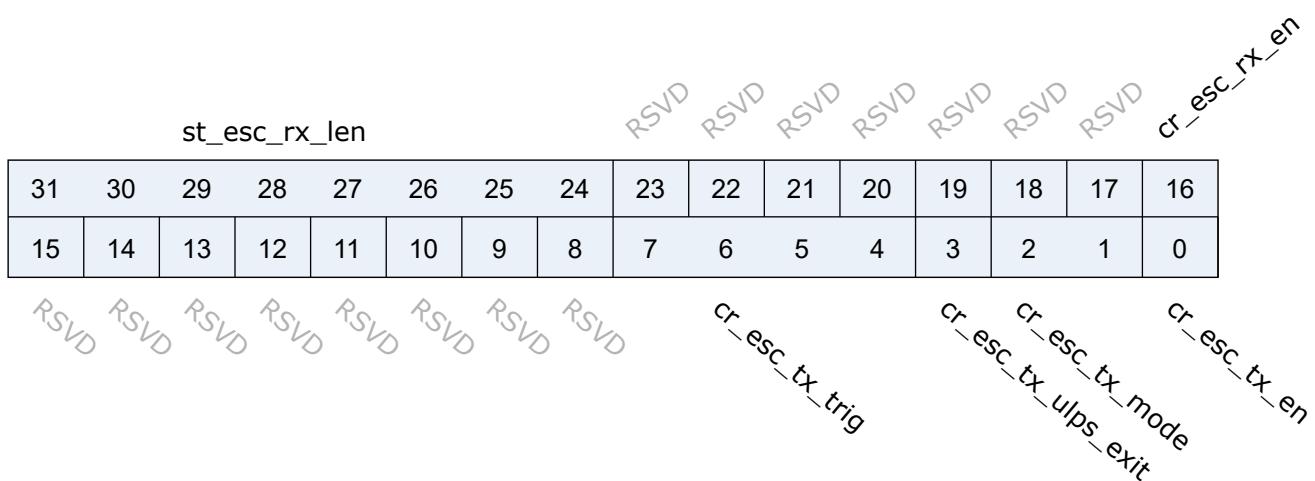
地址: 0x3001a100



位	名称	权限	复位值	描述
31:24	cr_hstx_vsa	r/w	8'd2	HSTX Vertical Sync Active width
23:16	cr_hstx_vfp	r/w	8'd2	HSTX Vertical Front Porch width
15:14	cr_vc	r/w	2'd0	Virtual Channel number
13:8	cr_dt	r/w	6'h3E	Data Type 6'h2C: YUV422, 8-bit 6'h0E: RGB565 6'h2E: RGB666, loosely packed 6'h3E: RGB888 Others: Reserved
7:6	cr_lane_mux_sel	r/w	2'd0	Controls lane order 2'd0: lane3, lane2, lane1, lane0 (no lane is switched) 2'd1: lane2, lane1, lane3, lane0 2'd2: lane1, lane3, lane2, lane0 2'd3: lane3, lane1, lane2, lane0
5	RSVD			
4	cr_hstx_mode	r/w	1'b1	HSTX mode select 1'b0: Sync Event mode 1'b1: Sync Pulse mode
3	RSVD			
2:1	cr_lane_num	r/w	2'd0	Lane number configuration 2'd0: 1-lane MIPI TX 2'd1: 2-lane MIPI TX 2'd2: 4-lane MIPI TX 2'd3: Reserved
0	cr_hstx_en	r/w	1'b0	HSTX function enable signal

## 12.4.2 dsi\_esc\_config

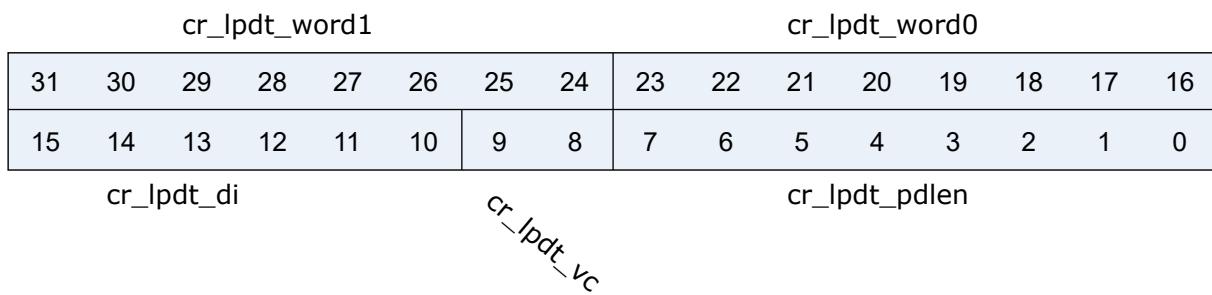
地址: 0x3001a104



位	名称	权限	复位值	描述
31:24	st_esc_rx_len	r	8'd0	Escape RX length received (Include packet header/data/footer) (unit: byte)
23:17	RSVD			
16	cr_esc_rx_en	r/w	1'b0	Escape RX enable signal
15:8	RSVD			
7:4	cr_esc_tx_trig	r/w	4'h0	Escape TX trigger command
3	cr_esc_tx_ulps_exit	w1p	1'b0	Escape TX ULPS exit signal trigger
2:1	cr_esc_tx_mode	r/w	2'd0	Escape TX mode select 2'd0: LPDT 2'd1: Trigger command 2'd2: ULPS Enable 2'd3: ULPS Disable
0	cr_esc_tx_en	w1p	1'b0	Escape TX enable signal

## 12.4.3 dsi\_lpdt\_tx\_config

地址: 0x3001a108



位	名称	权限	复位值	描述
31:24	cr_lpdt_word1	r/w	8'h0	LPDT TX word 1
23:16	cr_lpdt_word0	r/w	8'h0	LPDT TX word 0
15:10	cr_lpdt_di	r/w	6'h0	LPDT TX data identifier
9:8	cr_lpdt_vc	r/w	2'h0	LPDT TX virtual channel
7:0	cr_lpdt_pflen	r/w	8'd0	LPDT TX packet data length (exclude packet header & footer) (unit: byte)

#### 12.4.4 dsi\_hstx\_config

地址: 0x3001a10c

cr_hstx_out_th															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_hstx_pc															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

位	名称	权限	复位值	描述
31:27	RSVD			
26:16	cr_hstx_out_th	r/w	11'd840	<p>Line buffer threshold for controller to start transmitting each line (unit: pixel)            Formula: th = ceil( W * (1 - Fdp*BPP/Fhs/Ln) )            th: cr_hstx_out_th            W: Frame width            Fdp: Display (dp_dvp_tsrc) clock rate            Fhs: DSI byte clock rate (dsi_bit_clk/8 or mipipll_clk/8)            BPP: Byte-per-pixel (equals 3 for RGB888 &amp; RGB666; equals 2 for RGB565 &amp; YUV422_8)            Ln: DSI lane number (controlled by cr_lane_num)            Note: The minimum value is 6 for synchronization concern (Set to 6 if the formula result is negative or less than 6)</p>
15:11	RSVD			

位	名称	权限	复位值	描述
10:0	cr_hstx_pc	r/w	11'd1280	Pixel count of each line (frame width) (unit:pixel) Note: Pixel count should not exceed 1280 (720p) and should be a multiple of 4

#### 12.4.5 dsi\_int\_status

地址: 0x3001a110

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

dsi\_int\_status

位	名称	权限	复位值	描述
31:14	RSVD			
13:0	dsi_int_status	r	14'h0	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

#### 12.4.6 dsi\_int\_mask

地址: 0x3001a114

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

dsi\_int\_mask

位	名称	权限	复位值	描述
31:14	RSVD			
13:0	dsi_int_mask	r/w	14'h3FFF	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

#### 12.4.7 dsi\_int\_clear

地址: 0x3001a118

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

dsi\_int\_clear

位	名称	权限	复位值	描述
31:14	RSVD			
13:0	dsi_int_clear	w1p	14'h0	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

## 12.4.8 dsi\_int\_enable

地址: 0x3001a11c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

dsi int enable

位	名称	权限	复位值	描述
31:14	RSVD			
13:0	dsi_int_enable	r/w	14'h3FFF	<ul style="list-style-type: none"> <li>[13]: FIFO Error (check 0x60[7:4] for detail)</li> <li>[12]: Pixel Count Too Large Error</li> <li>[11]: Pixel Count Too Small Error</li> <li>[10]: Buffer Underrun Error</li> <li>[9]: Buffer Overrun Error</li> <li>[8]: RX LPDT FIFO ready</li> <li>[7]: TX LPDT FIFO ready</li> <li>[6:3]: RX Trigger Command</li> <li>[2]: RX ULPS Command</li> <li>[1]: RX LPDT End</li> <li>[0]: TX Escape Command End</li> </ul>

## 12.4.9 dsi\_fifo\_config\_0

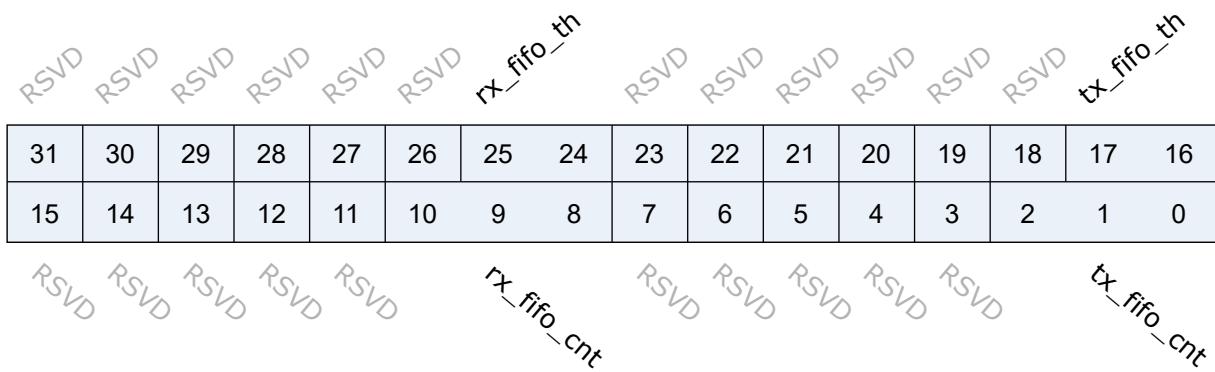
地址: 0x3001a160

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

位	名称	权限	复位值	描述
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1p	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1p	1'b0	Clear signal of TX FIFO
1	dsi_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	dsi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

#### 12.4.10 dsi\_fifo\_config\_1

地址: 0x3001a164



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:26	RSVD			
25:24	rx_fifo_th	r/w	2'd0	RX FIFO threshold, dma_rx_req will not be asserted if tx_fifo_cnt is less than this value
23:18	RSVD			
17:16	tx_fifo_th	r/w	2'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:11	RSVD			
10:8	rx_fifo_cnt	r	3'd0	RX FIFO available count

位	名称	权限	复位值	描述
7:3	RSVD			
2:0	tx_fifo_cnt	r	3'd4	TX FIFO available count

#### 12.4.11 dsi\_fifo\_wdata

地址: 0x3001a168

dsi\_fifo\_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dsi\_fifo\_wdata

位	名称	权限	复位值	描述
31:0	dsi_fifo_wdata	w	x	

#### 12.4.12 dsi\_fifo\_rdata

地址: 0x3001a16c

dsi\_fifo\_rdata

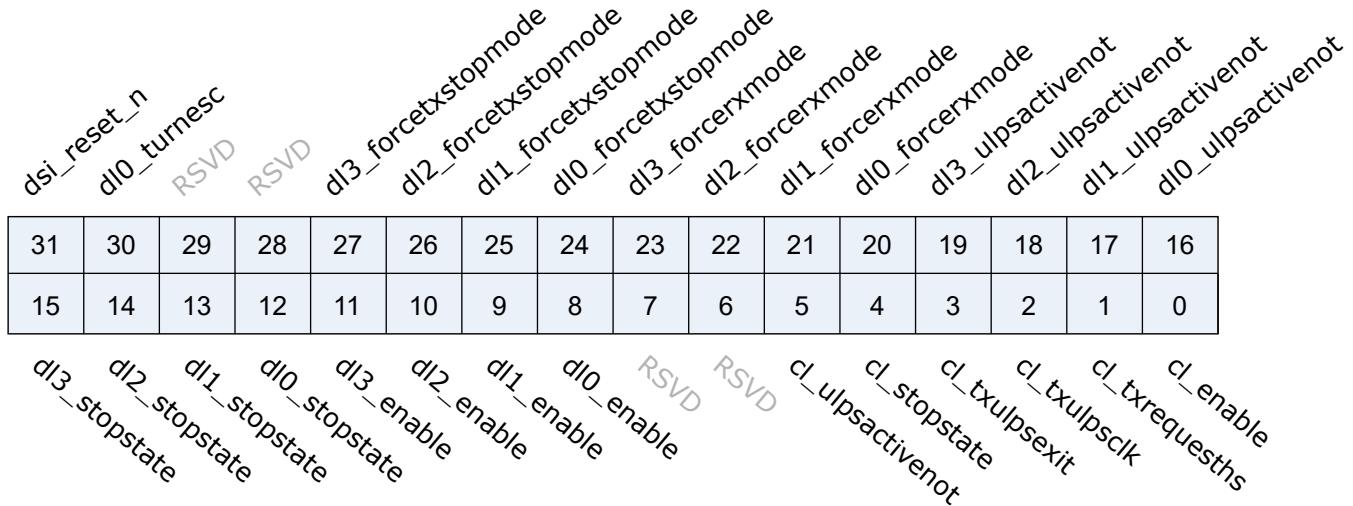
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dsi\_fifo\_rdata

位	名称	权限	复位值	描述
31:0	dsi_fifo_rdata	r	32'h0	

### 12.4.13 dphy\_config\_0

地址: 0x3001a180

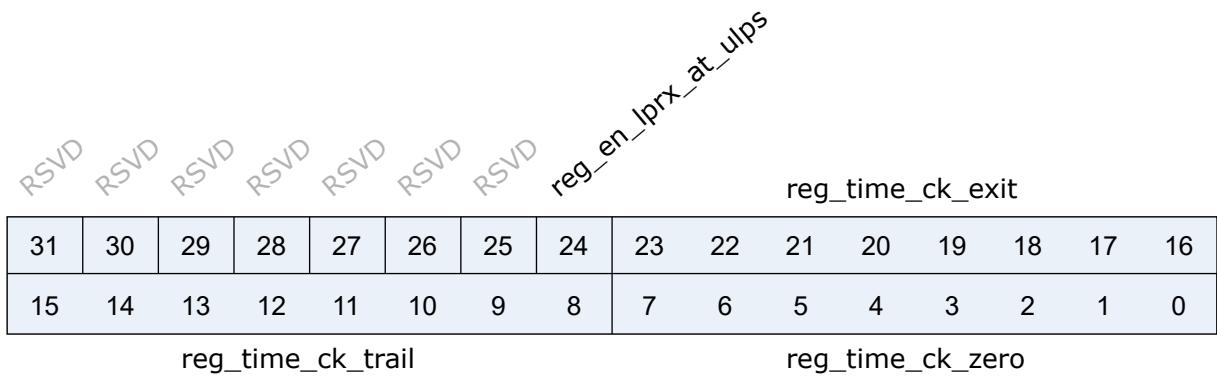


位	名称	权限	复位值	描述
31	dsi_reset_n	r/w	1'b0	MIPI DSI D-PHY reset pin
30	dl0_turnesc	w1p	1'b0	Data lane0 Bus Turnaround
29:28	RSVD			
27	dl3_forcetxstopmode	r/w	1'b0	Force Lane3 to Generate Stop State
26	dl2_forcetxstopmode	r/w	1'b0	Force Lane2 to Generate Stop State
25	dl1_forcetxstopmode	r/w	1'b0	Force Lane1 to Generate Stop State
24	dl0_forcetxstopmode	r/w	1'b0	Force Lane0 to Generate Stop State
23	dl3_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane3 immediately transitions to receive mode.
22	dl2_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane2 immediately transitions to receive mode.
21	dl1_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane1 immediately transitions to receive mode.
20	dl0_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane0 immediately transitions to receive mode.
19	dl3_ulpsactivenot	r	1'b1	Data lane3 is NOT in the ULP state
18	dl2_ulpsactivenot	r	1'b1	Data lane2 is NOT in the ULP state

位	名称	权限	复位值	描述
17	dl1_ulpsactivenot	r	1'b1	Data lane1 is NOT in the ULP state
16	dl0_ulpsactivenot	r	1'b1	Data lane0 is NOT in the ULP state
15	dl3_stopstate	r	1'b1	Data lane3 is in Stop state
14	dl2_stopstate	r	1'b1	Data lane2 is in Stop state
13	dl1_stopstate	r	1'b1	Data lane1 is in Stop state
12	dl0_stopstate	r	1'b1	Data lane0 is in Stop state
11	dl3_enable	r/w	1'b0	Data lane3 enable
10	dl2_enable	r/w	1'b0	Data lane2 enable
9	dl1_enable	r/w	1'b0	Data lane1 enable
8	dl0_enable	r/w	1'b0	Data lane0 enable
7:6	RSVD			
5	cl_ulpsactivenot	r	1'b1	Clock lane is NOT in the ULP state
4	cl_stopstate	r	1'b1	Clock lane is in Stop state
3	cl_txulpsexit	w1p	1'b0	Clock lane Transmit ULP Exit Sequence
2	cl_txulpsclk	r/w	1'b0	Clock lane Transmit Ultra-Low Power State
1	cl_txrequesths	r/w	1'b0	Clock lane High-Speed Transmit Request
0	cl_enable	r/w	1'b0	Clock lane enable

#### 12.4.14 dphy\_config\_1

地址: 0x3001a184



位	名称	权限	复位值	描述
31:25	RSVD			

位	名称	权限	复位值	描述
24	reg_en_lprx_at_ulps	r/w	1'b0	MIPI DSI D-PHY control register - reg_en_lprx_at_ulps
23:16	reg_time_ck_exit	r/w	8'h5	MIPI DSI D-PHY control register - reg_time_ck_exit (tx_clk_esc) txclkes: 40M, datarate: 800Mbps
15:8	reg_time_ck_trail	r/w	8'h3	MIPI DSI D-PHY control register - reg_time_ck_trail (tx_clk_esc)
7:0	reg_time_ck_zero	r/w	8'hF	MIPI DSI D-PHY control register - reg_time_ck_zero (tx_clk_esc)

#### 12.4.15 dphy\_config\_2

地址: 0x3001a188

reg_time_hs_exit								reg_time_hs_prep							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reg_time_hs_trail								reg_time_hs_zero							

位	名称	权限	复位值	描述
31:24	reg_time_hs_exit	r/w	8'h5	MIPI DSI D-PHY control register - reg_time_hs_exit (tx_clk_esc)
23:16	reg_time_hs_prep	r/w	8'h2	MIPI DSI D-PHY control register - reg_time_hs_prep (tx_clk_esc)
15:8	reg_time_hs_trail	r/w	8'h3	MIPI DSI D-PHY control register - reg_time_hs_trail (tx_clk_esc)
7:0	reg_time_hs_zero	r/w	8'h5	MIPI DSI D-PHY control register - reg_time_hs_zero (tx_clk_esc)

#### 12.4.16 dphy\_config\_3

地址: 0x3001a18c

reg_time_lpx								reg_time_reqrdy							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reg_time_ta_get								reg_time_ta_go							

位	名称	权限	复位值	描述
31:24	reg_time_lpx	r/w	8'h3	MIPI DSI D-PHY control register - reg_time_lpx (tx_clk_esc)
23:16	reg_time_reqrdy	r/w	8'h0	MIPI DSI D-PHY control register - reg_time_reqrdy
15:8	reg_time_ta_get	r/w	8'hF	MIPI DSI D-PHY control register - reg_time_ta_get (tx_clk_esc)
7:0	reg_time_ta_go	r/w	8'hC	MIPI DSI D-PHY control register - reg_time_ta_go (tx_clk_esc)

#### 12.4.17 dphy\_config\_4

地址: 0x3001a190

RSVD	RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	reg_time_wakeup	

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	reg_time_wakeup	r/w	16'h9C41	MIPI DSI D-PHY control register - reg_time_wakeup (tx_clk_esc)

#### 12.4.18 dphy\_config\_5

地址: 0x3001a194

reg_trig0_code								reg_trig1_code							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reg_trig2_code								reg_trig3_code							

位	名称	权限	复位值	描述
31:24	reg_trig0_code	r/w	8'b0100_0110	MIPI DSI D-PHY control register - reg_trig0_code

位	名称	权限	复位值	描述
23:16	reg_trig1_code	r/w	8'b1011_- 1010	MIPI DSI D-PHY control register - reg_trig1_- code
15:8	reg_trig2_code	r/w	8'b1000_- 0100	MIPI DSI D-PHY control register - reg_trig2_- code
7:0	reg_trig3_code	r/w	8'b0000_- 0101	MIPI DSI D-PHY control register - reg_trig3_- code

#### 12.4.19 dphy\_config\_6

地址: 0x3001a198

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

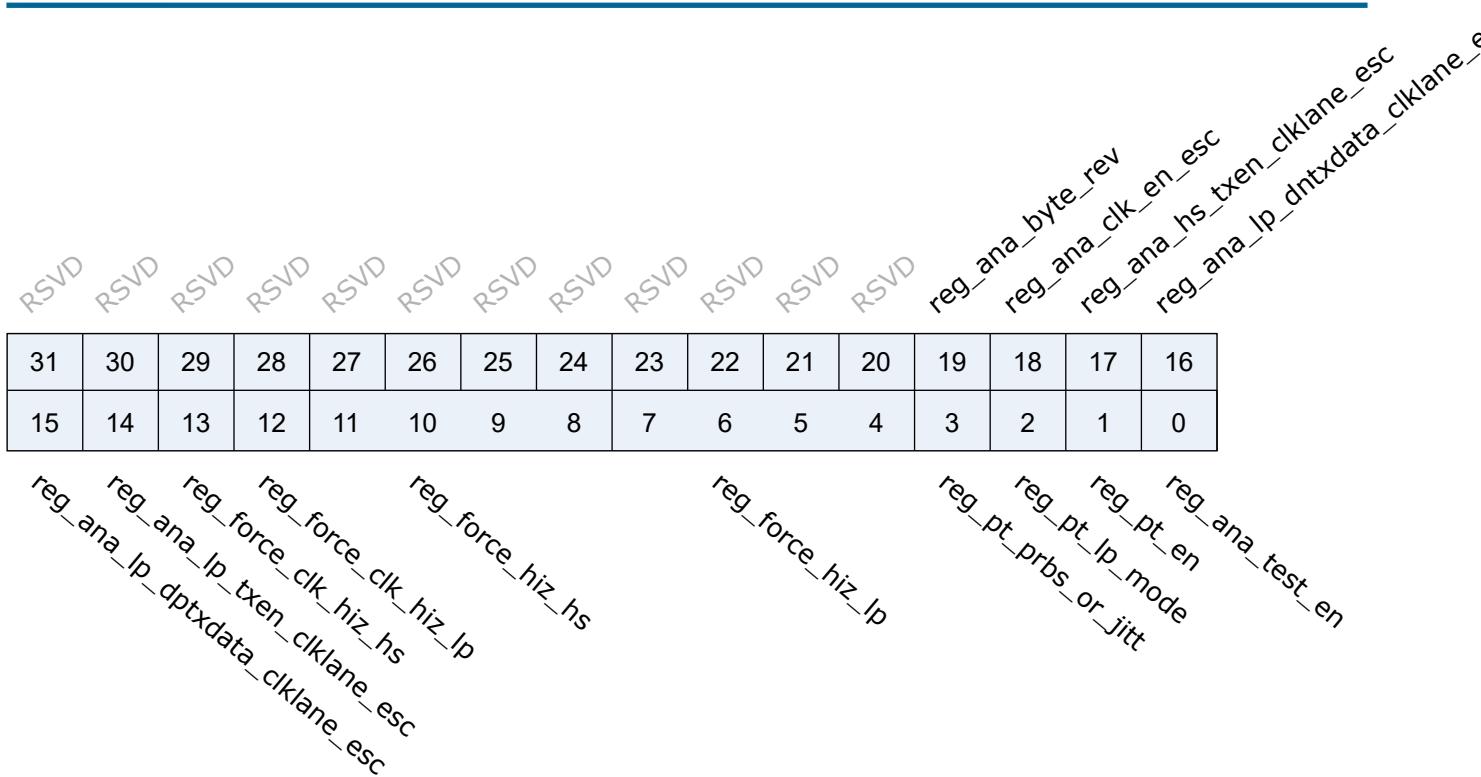
reg\_lpdt\_code

reg\_ulps\_code

位	名称	权限	复位值	描述
31:16	RSVD			
15:8	reg_lpdt_code	r/w	8'b1000_- 0111	MIPI DSI D-PHY control register - reg_lpdt_- code
7:0	reg_ulps_code	r/w	8'b0111_- 1000	MIPI DSI D-PHY control register - reg_ulps_- code

#### 12.4.20 dphy\_config\_7

地址: 0x3001a19c

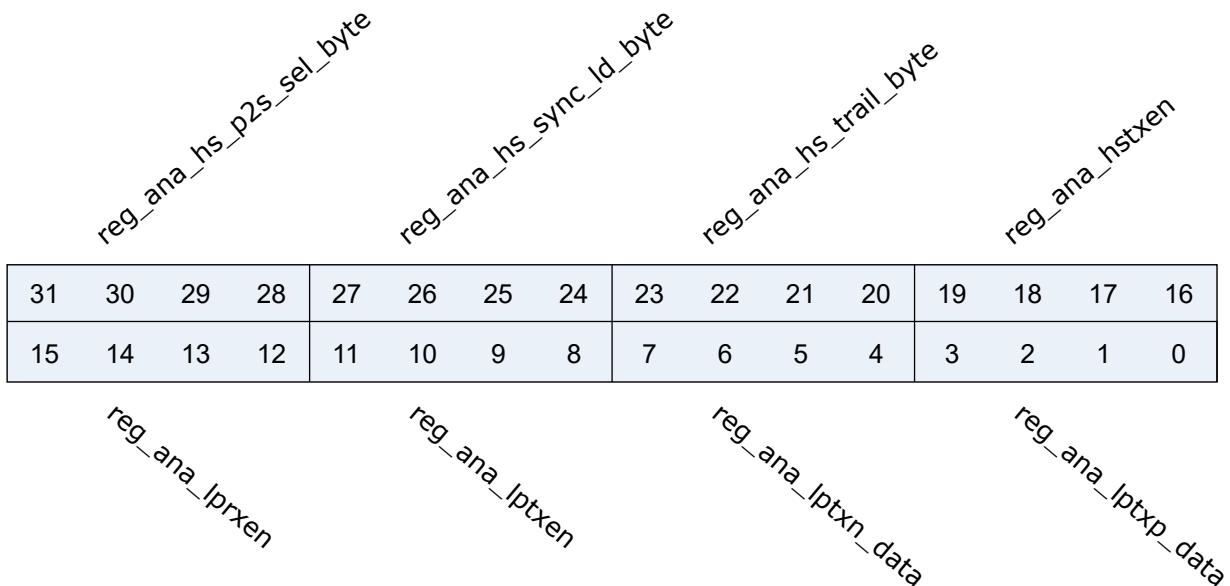


位	名称	权限	复位值	描述
31:20	RSVD			
19	reg_ana_byte_rev	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_byte_rev
18	reg_ana_clk_en_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_clk_en_esc
17	reg_ana_hs_txen_clklane_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_hs_txen_clklane_esc
16	reg_ana_lp_dntxdata_clklane_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_lp_dntxdata_clklane_esc
15	reg_ana_lp_dptxdata_clklane_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_lp_dptxdata_clklane_esc
14	reg_ana_lp_txen_clklane_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_lp_txen_clklane_esc
13	reg_force_clk_hiz_hs	r/w	1'b0	MIPI DSI D-PHY control register - reg_force_clk_hiz_hs
12	reg_force_clk_hiz_lp	r/w	1'b0	MIPI DSI D-PHY control register - reg_force_clk_hiz_lp
11:8	reg_force_hiz_hs	r/w	4'h0	MIPI DSI D-PHY control register - reg_force_hiz_hs

位	名称	权限	复位值	描述
7:4	reg_force_hiz_lp	r/w	4'h0	MIPI DSI D-PHY control register - reg_force_hiz_lp
3	reg_pt_prbs_or_jitt	r/w	1'b0	MIPI DSI D-PHY control register - reg_pt_prbs_or_jitt
2	reg_pt_lp_mode	r/w	1'b0	MIPI DSI D-PHY control register - reg_pt_lp_mode
1	reg_pt_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_pt_en
0	reg_ana_test_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_test_en

#### 12.4.21 dphy\_config\_8

地址: 0x3001a1a0



位	名称	权限	复位值	描述
31:28	reg_ana_hs_p2s_sel_byte	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hs_p2s_sel_byte
27:24	reg_ana_hs_sync_id_byte	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hs_sync_id_byte
23:20	reg_ana_hs_trail_byte	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hs_trail_byte
19:16	reg_ana_hstxen	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hstxen

位	名称	权限	复位值	描述
15:12	reg_ana_lprxen	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lprxen
11:8	reg_ana_lptxen	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lptxen
7:4	reg_ana_lptxn_data	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lptxn_data
3:0	reg_ana_lptxp_data	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lptxp_data

#### 12.4.22 dphy\_config\_9

地址: 0x3001a1a4

reg\_ana\_hs\_data\_out\_byte

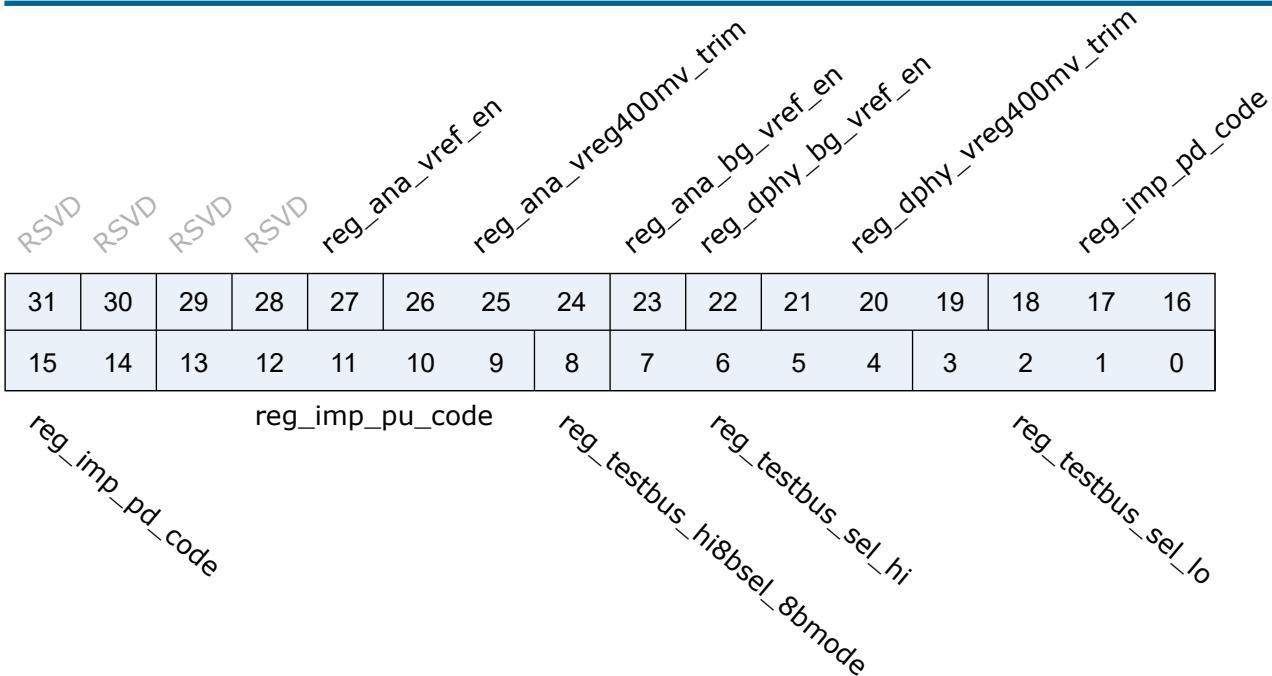
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_ana\_hs\_data\_out\_byte

位	名称	权限	复位值	描述
31:0	reg_ana_hs_data_out_byte	r/w	32'h0	MIPI DSI D-PHY control register - reg_ana_hs_data_out_byte

#### 12.4.23 dphy\_config\_10

地址: 0x3001a1a8



位	名称	权限	复位值	描述
31:28	RSVD			
27	reg_ana_vref_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_vref_en
26:24	reg_ana_vreg400mv_trim	r/w	3'h0	MIPI DSI D-PHY control register - reg_ana_vreg400mv_trim
23	reg_ana_bg_vref_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_bg_vref_en
22	reg_dphy_bg_vref_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_dphy_bg_vref_en
21:19	reg_dphy_vreg400mv_trim	r/w	3'h0	MIPI DSI D-PHY control register - reg_dphy_vreg400mv_trim
18:14	reg_imp_pd_code	r/w	5'h9	MIPI DSI D-PHY control register - reg_imp_pd_code
13:9	reg_imp_pu_code	r/w	5'h8	MIPI DSI D-PHY control register - reg_imp_pu_code
8	reg_testbus_hi8bsel_8bmode	r/w	1'b0	MIPI DSI D-PHY control register - reg_testbus_hi8bsel_8bmode
7:4	reg_testbus_sel_hi	r/w	4'h0	MIPI DSI D-PHY control register - reg_testbus_sel_hi
3:0	reg_testbus_sel_lo	r/w	4'h0	MIPI DSI D-PHY control register - reg_testbus_sel_lo

### 12.4.24 dphy\_config\_11

地址: 0x3001a1ac

`reg_dsi_ana_1`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

`reg_dsi_ana_0`

位	名称	权限	复位值	描述
31:16	<code>reg_dsi_ana_1</code>	r/w	16'h0	MIPI DSI D-PHY control register - <code>reg_dsi_ana_1</code>
15:0	<code>reg_dsi_ana_0</code>	r/w	16'hc14	MIPI DSI D-PHY control register - <code>reg_dsi_ana_0</code>

### 12.4.25 dphy\_config\_12

地址: 0x3001a1b0

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

`reg_dsi_ana_2`

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	<code>reg_dsi_ana_2</code>	r/w	16'h0	MIPI DSI D-PHY control register - <code>reg_dsi_ana_2</code>

### 12.4.26 dphy\_config\_13

地址: 0x3001a1b4

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

`reg_rd_dig_test_bus`

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	reg_rd_dig_test_bus	r	16'h0	MIPI DSI D-PHY control register - reg_rd_dig_test_bus

#### 12.4.27 dphy\_config\_14

地址: 0x30001a1b8

reg\_pt\_free\_rep\_pat

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_pt\_free\_rep\_pat

位	名称	权限	复位值	描述
31:0	reg_pt_free_rep_pat	r/w	32'h8765432	MIPI DSI D-PHY control register - reg_pt_free_rep_pat

#### 12.4.28 dphy\_config\_15

地址: 0x30001a1bc

RSVD	RSVD	reg_csi_rst_n_pre	RSVD	RSVD	RSVD	RSVD	RSVD	reg_mipi_ldo_fast	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	reg_dphy_ldo11_rfb_sw
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD	RSVD	RSVD	reg_dphy_short_ldo11	RSVD	RSVD	RSVD	RSVD	reg_dphy_pu_ldo11	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	reg_dsi_dc_tp_out_en
31:30															

位	名称	权限	复位值	描述
31:30	RSVD			

位	名称	权限	复位值	描述
29	reg_csi_RST_N_Pre	r/w	1'b0	Note: reg_csi_RST_N_Pre should be released at least 2 us BEFORE releasing csi_reset_n
28	reg_dsi_RST_N_Pre	r/w	1'b0	Note: reg_dsi_RST_N_Pre should be released at least 2 us BEFORE releasing dsi_reset_n
27:25	RSVD			
24	reg_mipi_ldo_fast	r/w	1'b0	
23:21	RSVD			
20	reg_ten_dsi_ldo	r/w	1'b0	
19	RSVD			
18:16	reg_dphy_ldo11_rfb_sw	r/w	3'd4	
15:13	RSVD			
12	reg_dphy_short_ldo11	r/w	1'b0	
11:9	RSVD			
8	reg_dphy_pu_ldo11	r/w	1'b1	enable LDO11 for both dsi and csi
7:5	RSVD			
4	reg_dsi_pw_avdd1815	r/w	1'b0	0: power switch on
3:1	RSVD			
0	reg_dsi_dc_tp_out_en	r/w	1'b0	

### 12.4.29 dphy\_config\_16

地址: 0x3001a1c0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD

reg\_dsi\_byte\_clk\_inv  
reg\_dsi\_lpx\_clk\_inv

位	名称	权限	复位值	描述
31:2	RSVD			
1	reg_dsi_lprx_clk_inv	r/w	1'b0	
0	reg_dsi_byte_clk_inv	r/w	1'b0	

### 12.4.30 dummy\_reg

地址: 0x3001a1fc

dummy\_reg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dummy\_reg

位	名称	权限	复位值	描述
31:0	dummy_reg	r/w	32'h0	Dummy registers

## 13.1 简介

CAM(Camera) 模块负责将并行接口 (DVP) 转换成一般总线接口 (AHB)，把图像传感器生成的像素数据以平面或打包格式写入系统内存中，作为后续影像传输或压缩使用。CAM 模块拥有灵活的输出格式配置，可以满足多种多样的图像处理需求。

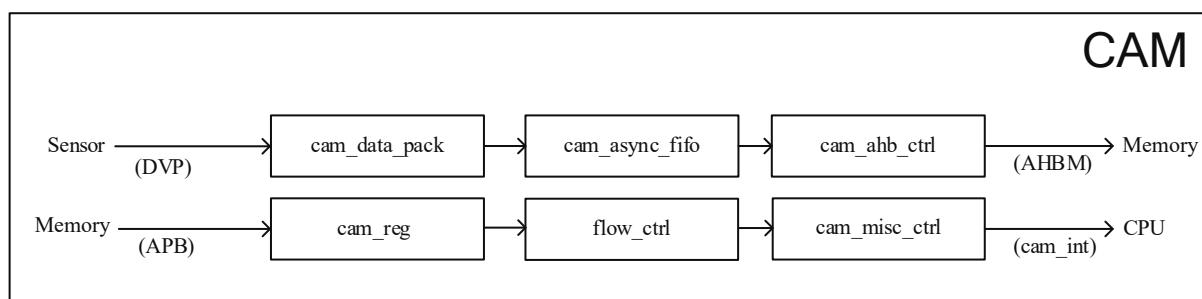


图 13.1: Cam 框图

## 13.2 主要特征

- 并行接口 8-bitDVP 信号，高速数据传输 (80M)，可配置 DVP 信号有效水平与逻辑组合
- 支持 8-bit/16-bit/24-bit 输入像素位宽
- 支持多达 36 种数据输入源
- 支持将 RGB888 输入格式转为 RGB565 或 RGBA8888 输出
- 支持影片模式和照片模式
- 可配置丢弃模式，包括：
  - 丢弃奇数位数据
  - 丢弃偶数位数据
  - 丢弃奇数行的奇数位数据
  - 丢弃奇数行的偶数位数据
- 可配置的图像传感器行帧同步信号选择和极性选择

- 支持图像矩形裁剪
- 支持行帧同步信号的完整性检测
- AHB 总线通信接口
- 512 字节缓存 FIFO 以应对总线偶而忙碌的状态
- 连续缓存多达 4 组图片信息
- 多种应用中断，有利于弹性使用与出错提示

## 13.3 功能描述

### 13.3.1 DVP(Digital Video Port) 信号与配置

DVP (Digital Video Port) 是并行接口，主要有时钟，帧同步，行同步与 8-bit 数据管脚，时钟的极限限制在 80MHz，所以一般用于 5MP 以下分辨率 Sensor。芯片内可独立配置帧同步与行同步的有效电平，并在有效数据上提供四种模式：

- A. 帧同步与行同步同时有效 (“&” 逻辑)
- B. 帧同步或行同步择一有效 (“|” 逻辑)
- C. 帧同步有效
- D. 行同步有效

### 13.3.2 YCbCr 格式

亮度信号被称为 Y，色度信号是由两个互相独立的信号组成。视颜色系统和格式不同，两种色度信号经常被称为 U、V 或 Pb、Pr 或 Cb、Cr。这是由不同的编码格式产生的，但实际上它们的概念基本相同。

由于人类视网膜上识别亮度的视网膜杆细胞要多于识别色度的视网膜锥细胞，人眼对亮度的敏感程度要高于对色度的敏感程度。所以可以将色度信息丢弃一部分而不被人眼所察觉。

色度信号分辨率最高的格式是 4:4:4，即每 4 点 Y 采样对应了 4 点 Cb 和 4 点 Cr 采样。而 4:2:2 是每 4 点 Y 采样对应了 2 点 Cb 和 2 点 Cr 采样，在这种格式中，色度信号的扫描线数量和亮度信号一样多，但是每条扫描线上的色度采样点却只有亮度信号的一半。与上面提到的格式不同，4:2:0 并不是每 4 点 Y 采样对应 2 点 Cb 和 0 点 Cr 采样，而是每 4 点 Y 采样对应 1 点 Cb 和 1 点 Cr 采样。4:0:0 是丢弃所有的色度信息，即灰度图。

### 13.3.3 输入源

通过配置可以选择 CAM 的数据输入源，配置值与输入源的对应关系如下表所示：

Value	Input Type	Data Width	Value	Input Type	Data Width
1	Active DVP(TG)	8-bit	22	Adjust_1	16-bit
2	Defect Correct	8-bit	23	Adjust_2	16-bit
3	CCM_R	8-bit	24	Adjust_3	16-bit
4	CCM_G	8-bit	25	YUV420_0	16-bit
5	CCM_B	8-bit	26	YUV420_1	16-bit
6	Gamma_R	8-bit	27	YUV420_2	16-bit
7	Gamma_G	8-bit	28	YUV420_3	16-bit
8	-	-	29	Gamma_B	8-bit
9	BNR	8-bit	30	WDR_Y	8-bit
10	NR	16-bit	31	WDR_U	8-bit
11	EE	16-bit	32	WDR_V	8-bit
17	Scaler_0	16-bit	33	LSC	8-bit
18	Scaler_1	16-bit	34	AWB2	16-bit
19	Scaler_2	16-bit	35	YUV2RGB	24-bit
20	Scaler_3	16-bit	36	DVP_AS_2X	16-bit
21	Adjust_0	16-bit			

图 13.2: 输入源选择

### 13.3.4 影片模式/照片模式

照片模式下当软件给的固定大小的存储器被写满时，CAM 会停止，需要软件进行 POP 操作将空间空出后才会继续写入。

影片模式下会在软件给的固定大小的存储器上不停地复写，也就是将内存当作 ring buffer 的概念，无需软件进行 POP 操作，使用上要确保图片被实时取出或是跟着 MJPEG 模块做连动。

### 13.3.5 图像矩形裁剪

通过寄存器 HSYNC\_CONTROL 和 VSYNC\_CONTROL 的高 16 位和低 16 位分别设置行同步信号和帧同步信号裁剪的起始和结束位置，就可以将指定位置和大小的矩形窗口内的图像裁剪下来，超出矩形部分的数据会被丢弃。其中行同步信号起始和结束设置的是像素序号，帧同步信号起始和结束设置的是行号，裁剪后的图像包含起始点而不包含结束点。

### 13.3.6 行帧同步信号完整性检测

通过寄存器 FRAME\_SIZE\_CONTROL 的低 16 位和高 16 位可以分别设置行同步信号比较值和帧同步信号比较值，可以对信号的完整性进行检测。其中行同步信号设置的是每行的总像素数，帧同步信号设置的是总行数。当一帧图像的行或帧同步信号计数值与比较值不相等时，会有对应的中断产生。

### 13.3.7 缓存图片信息

模块内部包含 4 组 FIFO 记录图片地址和图片大小。每当此模块完整写入一帧到内存，便会将此帧图片的起始地址和图片大小纪录于此 FIFO 中，但要注意的是当发生内存剩余不足，或是 4 组 FIFO 满存的状况时，模块会自动丢掉接下来图片的讯息，在图片信息取出的部分，可通过 APB 接口做 pop 的动作，将最旧的图片信息空出，此时 FIFO 会自动推进，保证 FIFO 内部图片信息的时序，如下图：

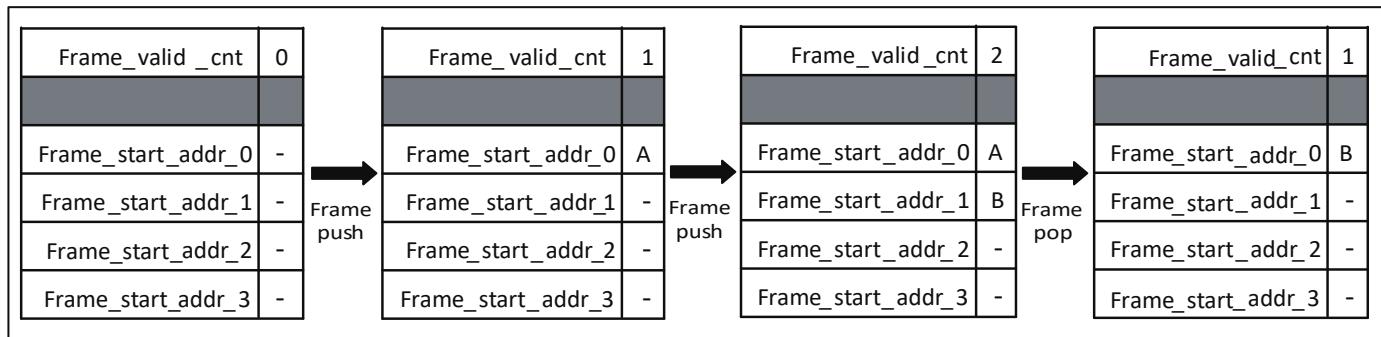


图 13.3: FIFO 框架

### 13.3.8 支持多种中断信息 (可独立开关配置)

A、Normal 中断-可设定固定写入几张图片后发出中断

B、Memory 中断-当内存被复写时，发出中断

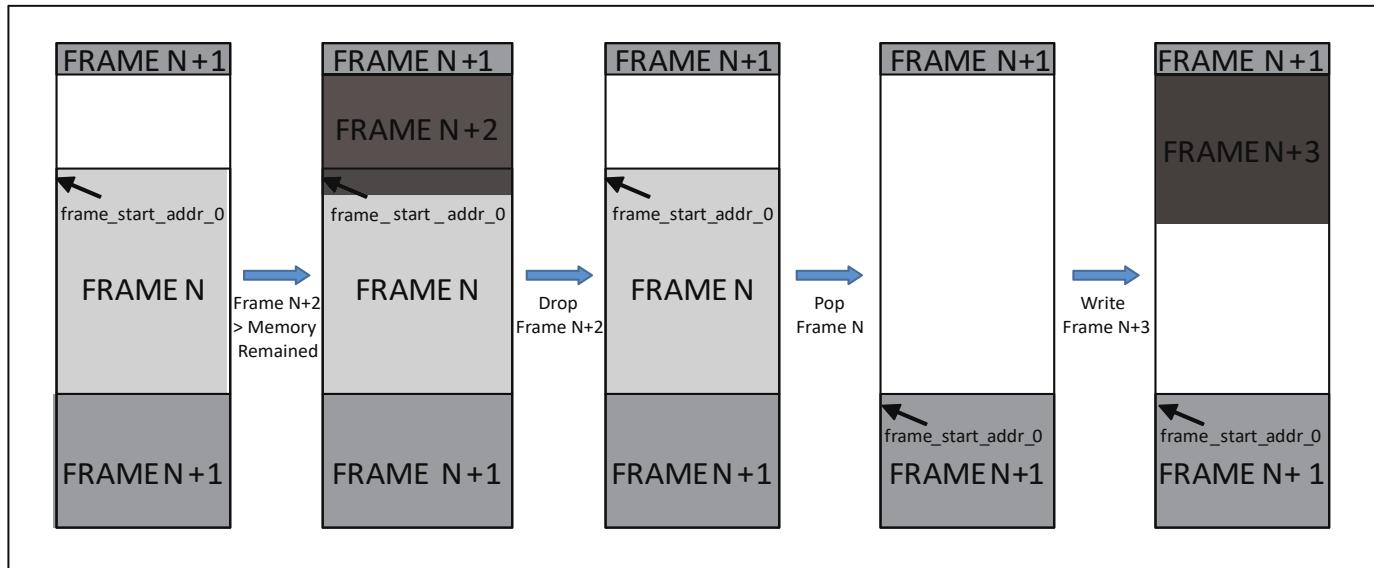


图 13.4: 内存

C、Frame 中断-当未处理图片超过 4 组时，发出中断

D、FIFO 中断-总线来不及写入内存导致 FIFO 溢出时，发出中断

E、Hsync 中断 - 当一帧图像中某行的像素点数与设置值不相等时，发出中断

F、Vsync 中断 - 当一帧图像的总行数与设置值不相等时，发出中断

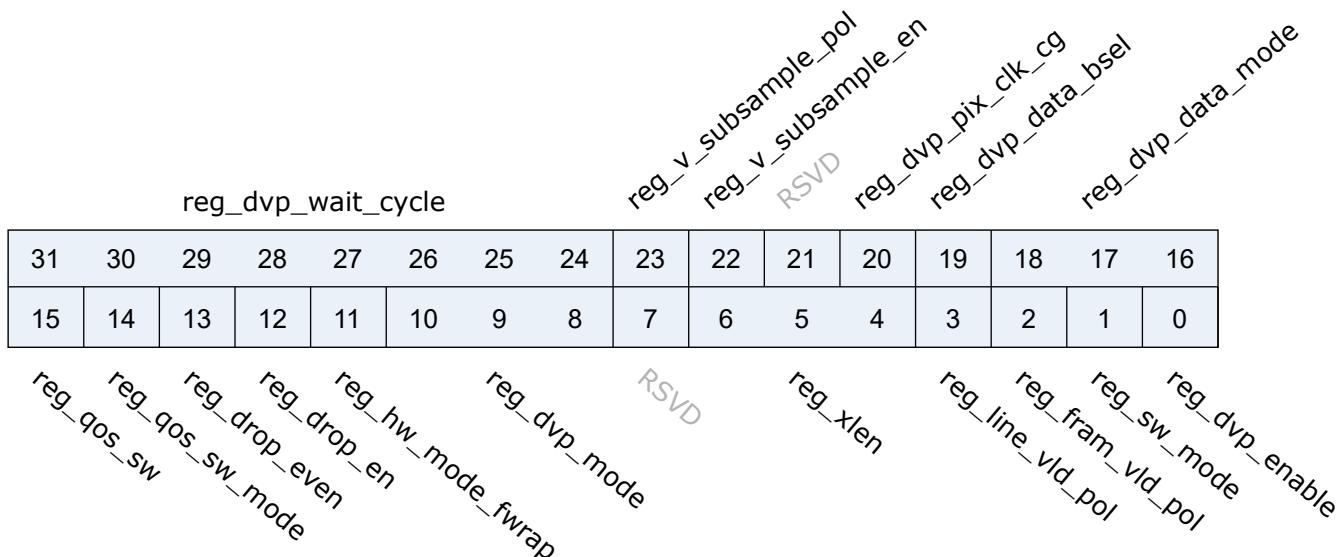
## 13.4 寄存器描述

名称	描述
dvp2axi_configue	
dvp2axi_addr_start	
dvp2axi_mem_bcnt	
dvp_status_and_error	
dvp2axi_frame_bcnt	
dvp_frame_fifo_pop	
dvp2axi_frame_vld	
dvp2axi_frame_period	

名称	描述
dvp2axi_misc	
dvp2axi_hsync_crop	
dvp2axi_vsync_crop	
dvp2axi_fram_exm	
frame_start_addr0	
frame_start_addr1	
frame_start_addr2	
frame_start_addr3	
frame_id_sts01	
frame_id_sts23	
dvp_debug	
dvp_dummy_reg	

### 13.4.1 dvp2axi\_configure

地址: 0x30012000



位	名称	权限	复位值	描述
31:24	reg_dvp_wait_cycle	r/w	8'h40	Cycles in FSM Wait mode

位	名称	权限	复位值	描述
23	reg_v_subsample_pol	r/w	1'b0	DVP2BUS vertical sub-sampling polarity 1'b0: Odd lines are masked 1'b1: Even lines are masked
22	reg_v_subsample_en	r/w	1'b0	DVP2BUS vertical sub-sampling enable
21	RSVD			
20	reg_dvp_pix_clk_cg	r/w	1'b0	DVP pix clk gate
19	reg_dvp_data_bsel	r/w	1'b0	Byte select signal for DVP 8-bit mode, don't care if reg_dvp_data_8bit is disabled 1'b0: Select the lower byte of pix_data 1'b1: Select the upper byte of pix_data
18:16	reg_dvp_data_mode	r/w	3'b0	DVP 8-bit mode enable 3'd0: DVP pix_data is 16-bit wide 3'd1: DVP pix_data is 24-bit mode 3'd2: DVP pix_data is 24-comp-16-bit mode 3'd3: DVP pix_data is 24-exp-32-bit mode 3'd4: DVP pix_data is 8-bit wide Others - Reserved
15	reg_qos_sw	r/w	1'b0	AXI Qos software mode value
14	reg_qos_sw_mode	r/w	1'b0	AXI QoS software mode enable
13	reg_drop_even	r/w	1'b0	Only effect when reg_drop_en=1 : 1'b1 : Drop all even bytes 1'b0 : Drop all odd bytes
12	reg_drop_en	r/w	1'b0	Drop mode Enable
11	reg_hw_mode_fwrap	r/w	1'b1	DVP2BUS HW mode with frame start address wrap to reg_addr_start
10:8	reg_dvp_mode	r/w	3'd0	Image sensor mode selection: 3'd0-Vsync&Hsync 3'd1-Vsync Hsync 3'd2-Vsync 3'd3-Hsync
7	RSVD			
6:4	reg_xlen	r/w	3'd3	burst length setting 3'd0 - Single / 3'd1 - INCR4 / 3'd2 - INCR8 3'd3 - INCR16 / 3'd5 - INCR32 / 3'd6 - INCR64
3	reg_line_vld_pol	r/w	1'b1	Image sensor line valid polarity, 1'b0 - Active low, 1'b1 - Active high

位	名称	权限	复位值	描述
2	reg_fram_vld_pol	r/w	1'b1	Image sensor frame valid polarity, 1'b0 - Active low, 1'b1 - Active high
1	reg_sw_mode	r/w	1'b0	DVP2BUS SW manual mode (don't care if reg_swap_mode is enabled)
0	reg_dvp_enable	r/w	1'b0	module enable

### 13.4.2 dvp2axi\_addr\_start

地址: 0x30012004

reg\_addr\_start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_addr\_start

位	名称	权限	复位值	描述
31:0	reg_addr_start	r/w	32'h80000000	AXI start address

### 13.4.3 dvp2axi\_mem\_bcnt

地址: 0x30012008

reg\_mem\_burst\_cnt

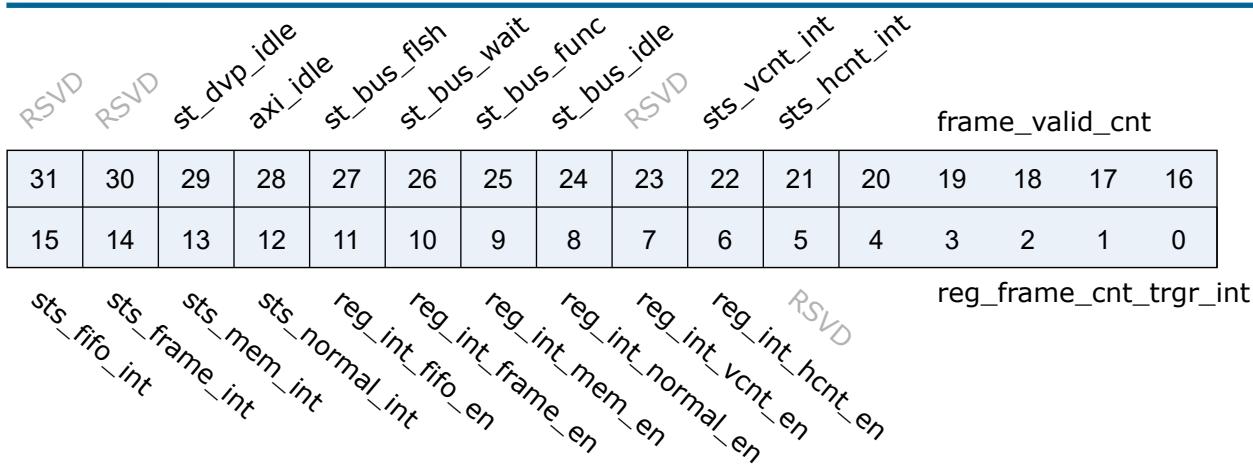
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_mem\_burst\_cnt

位	名称	权限	复位值	描述
31:0	reg_mem_burst_cnt	r/w	32'hC000	AXI burst cnt before wrap to "reg_addr_start"

### 13.4.4 dvp\_status\_and\_error

地址: 0x3001200c



位	名称	权限	复位值	描述
31:30	RSVD			
29	st_dvp_idle	r	1'b1	DVP2BUS asynchronous fifo idle status
28	axi_idle	r	1'b1	DVP2BUS AHB idle status
27	st_bus_fish	r	1'b0	DVP in flush state
26	st_bus_wait	r	1'b0	DVP in wait state
25	st_bus_func	r	1'b0	DVP in functional state
24	st_bus_idle	r	1'b1	DVP in idle state
23	RSVD			
22	sts_vcvt_int	r	1'b0	Vsync valid line count non-match interrupt status
21	sts_hcnt_int	r	1'b0	Hsync valid pixel count non-match interrupt status
20:16	frame_valid_cnt	r	5'd0	Frame counts in memory before read out in SW mode
15	sts_fifo_int	r	1'b0	FIFO OverWrite interrupt status
14	sts_frame_int	r	1'b0	Frame OverWrite interrupt status
13	sts_mem_int	r	1'b0	Memory OverWrite interrupt status
12	sts_normal_int	r	1'b0	Normal Write interrupt status
11	reg_int_fifo_en	r/w	1'b1	FIFO OverWrite interrupt enable
10	reg_int_frame_en	r/w	1'b0	Frame OverWrite interrupt enable
9	reg_int_mem_en	r/w	1'b0	Memory OverWrite interrupt enable
8	reg_int_normal_en	r/w	1'b0	Normal Write interrupt enable
7	reg_int_vcvt_en	r/w	1'b0	Vsync valid line count match interrupt enable

位	名称	权限	复位值	描述
6	reg_int_hcnt_en	r/w	1'b0	Hsync valid pixel count match interrupt enable
5	RSVD			
4:0	reg_frame_cnt_trgr_int	r/w	5'd0	Frame to issue interrupt at SW Mode

### 13.4.5 dvp2axi\_frame\_bcnt

地址: 0x30012010

reg_frame_byte_cnt															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:0	reg_frame_byte_cnt	r/w	32'h7e90	Single Frame byte cnt(Need pre-calculation)

### 13.4.6 dvp\_frame\_fifo\_pop

地址: 0x30012014

位	名称	权限	复位值	描述
31:10	RSVD			
9	reg_int_vcnt_clr	w1p	1'd0	Interrupt clear
8	reg_int_hcnt_clr	w1p	1'd0	Interrupt clear
7	reg_int_fifo_clr	w1p	1'd0	Interrupt clear
6	reg_int_frame_clr	w1p	1'd0	Interrupt clear

位	名称	权限	复位值	描述
5	reg_int_mem_clr	w1p	1'd0	Interrupt clear
4	reg_int_normal_clr	w1p	1'd0	Interrupt clear
3:1	RSVD			
0	rfifo_pop	w1p	1'b0	Write this bit will trigger fifo pop

### 13.4.7 dvp2axi\_frame\_vld

地址: 0x30012018

reg\_frame\_n\_vld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_frame\_n\_vld

位	名称	权限	复位值	描述
31:0	reg_frame_n_vld	r/w	32'hffff_-ffff	Bitwise frame valid in period

### 13.4.8 dvp2axi\_frame\_period

地址: 0x3001201c

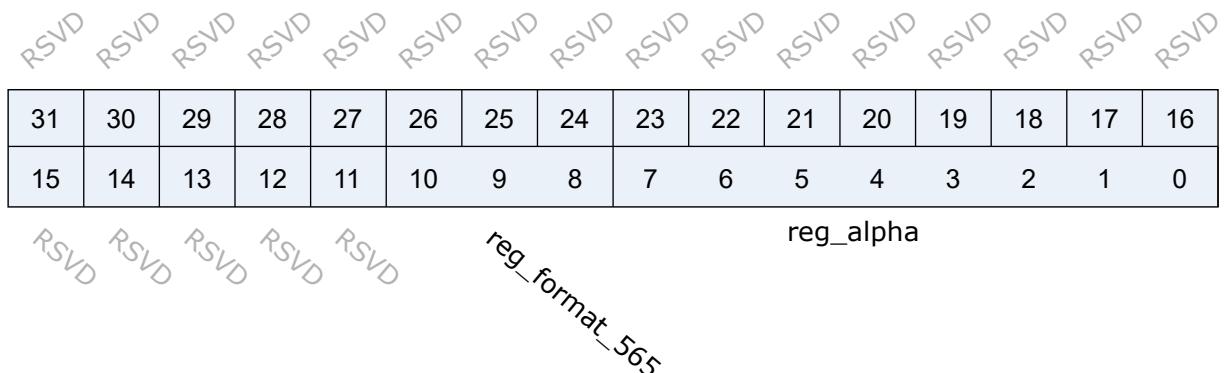
RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

reg\_frame\_period

位	名称	权限	复位值	描述
31:5	RSVD			
4:0	reg_frame_period	r/w	5'h0	Frame period cnt. (EX. Set this register 0, the period is 1)

### 13.4.9 dvp2axi\_misc

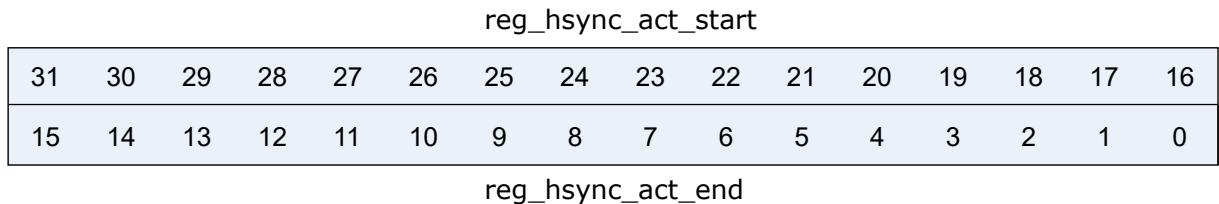
地址: 0x30012020



位	名称	权限	复位值	描述
31:11	RSVD			
10:8	reg_format_565	r/w	3'd0	Only work when reg_dvp_data_mode=2 (24-comp-16-bit mode) 3'd0: B2(5)B1(6)B0(5) 3'd1: B1(5)B2(6)B0(5) 3'd2: B2(5)B0(6)B1(5) 3'd3: B0(5)B2(6)B1(5) 3'd4: B1(5)B0(6)B2(5) 3'd5: B0(5)B1(6)B2(5)
7:0	reg_alpha	r/w	8'h0	Only work when "reg_dvp_data_mode==2'd3(DVP_pix_data is 24-exp-32-bit mode)" The value of [31:24]

### 13.4.10 dvp2axi\_hsync\_crop

地址: 0x30012030



位	名称	权限	复位值	描述
31:16	reg_hsync_act_start	r/w	16'h0	Valid hsync start cnt

位	名称	权限	复位值	描述
15:0	reg_hsync_act_end	r/w	16'hFFFF	Valid hsync end cnt

### 13.4.11 dvp2axi\_vsync\_crop

地址: 0x30012034

reg\_vsync\_act\_start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_vsync\_act\_end

位	名称	权限	复位值	描述
31:16	reg_vsync_act_start	r/w	16'h0	Valid vsync start cnt
15:0	reg_vsync_act_end	r/w	16'hFFFF	Valid vsync end cnt

### 13.4.12 dvp2axi\_fram\_exm

地址: 0x30012038

reg\_total\_vcnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg\_total\_hcnt

位	名称	权限	复位值	描述
31:16	reg_total_vcnt	r/w	16'h0	Total valid line count in a frame
15:0	reg_total_hcnt	r/w	16'h0	Total valid pix count in a line

### 13.4.13 frame\_start\_addr0

地址: 0x30012040

frame\_start\_addr\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame\_start\_addr\_0

位	名称	权限	复位值	描述
31:0	frame_start_addr_0	r	32'd0	DVP2BUS PIC 0 Start address

#### 13.4.14 frame\_start\_addr1

地址: 0x30012048

frame\_start\_addr\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame\_start\_addr\_1

位	名称	权限	复位值	描述
31:0	frame_start_addr_1	r	32'd0	DVP2BUS PIC 1 Start address

#### 13.4.15 frame\_start\_addr2

地址: 0x30012050

frame\_start\_addr\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame\_start\_addr\_2

位	名称	权限	复位值	描述
31:0	frame_start_addr_2	r	32'd0	DVP2BUS PIC 2 Start address

#### 13.4.16 frame\_start\_addr3

地址: 0x30012058

frame\_start\_addr\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame\_start\_addr\_3

位	名称	权限	复位值	描述
31:0	frame_start_addr_3	r	32'd0	DVP2BUS PIC 3 Start address

### 13.4.17 frame\_id\_sts01

地址: 0x30012060

frame\_id\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame\_id\_0

位	名称	权限	复位值	描述
31:16	frame_id_1	r	16'd0	DVP2BUS PIC 1 ID
15:0	frame_id_0	r	16'd0	DVP2BUS PIC 0 ID

### 13.4.18 frame\_id\_sts23

地址: 0x30012064

frame\_id\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame\_id\_2

位	名称	权限	复位值	描述
31:16	frame_id_3	r	16'd0	DVP2BUS PIC 3 ID
15:0	frame_id_2	r	16'd0	DVP2BUS PIC 2 ID

### 13.4.19 dvp\_debug

地址: 0x300120f0

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD	RSVD	RSVD	RSVD	reg_id_latch_line	RSVD	RSVD	RSVD	RSVD	reg_dvp_dbg_sel	reg_dvp_dbg_en						
------	------	------	------	-------------------	------	------	------	------	-----------------	----------------	--	--	--	--	--	--

位	名称	权限	复位值	描述
31:12	RSVD			
11:8	reg_id_latch_line	r/w	4'd5	ID latch timing (line count)
7:4	RSVD			
3:1	reg_dvp_dbg_sel	r/w	3'd0	DVP2BUS debug flag selection
0	reg_dvp_dbg_en	r/w	1'b0	DVP2BUS debug flag enable

### 13.4.20 dvp\_dummy\_reg

地址: 0x300120fc

RESERVED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RESERVED

位	名称	权限	复位值	描述
31:0	RESERVED	rsvd	32'hf0f0f0f0	RESERVED

# 14

IR

## 14.1 简介

红外遥控（Infrared remote，简称 IR）是一种无线、非接触式控制技术，具有抗干扰能力强、信息传输可靠、功耗低、成本低等优点。红外遥控的发射电路是采用红外发光二极管来发出经过调制的红外光波；接收电路由红外接收二极管、三极管或硅光电池组成，它们将红外发射器发射的红外光转换为相应的电信号，再送至后置放大器。

## 14.2 主要特征

- 支持以固定协议 NEC、RC-5 接收数据
- 支持以脉冲宽度计数方式接收任意格式数据
- 强大的红外波形编辑能力，可发出符合各种协议的波形
- 多达 15 个档位的功率设定，以适应不同的功耗要求
- 接收最多支持 64-bit 数据位
- 发送在非自由模式下最多支持 128-bit 数据位，自由模式下可连续发送任意长度数据
- 4\*4 字节的发送 FIFO，64\*2 字节的接收 FIFO
- 自由模式下发送 FIFO 宽度可调，支持 8/16/24/32-bit
- 载波频率与占空比可编程
- 最高工作频率为 32MHz
- 发送支持 DMA 模式
- 发送和接收结束中断

## 14.3 功能描述

### 14.3.1 固定协议接收

IR 接收支持两种固定的协议，分别为 NEC 协议和 RC-5 协议。

- NEC 协议

NEC 协议的逻辑 1 与逻辑 0 波形如下图所示：

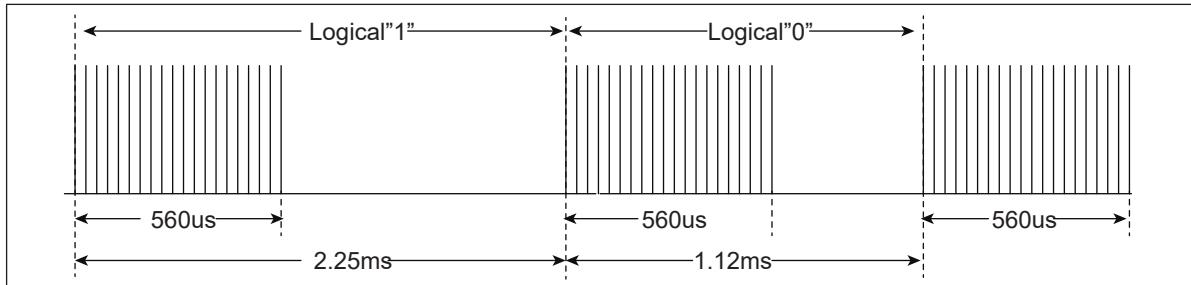


图 14.1: NEC 逻辑波形

逻辑 1 为 2.25ms，脉冲时间 560us；逻辑 0 位 1.12ms，脉冲时间 560us。NEC 协议的具体格式如下图所示：

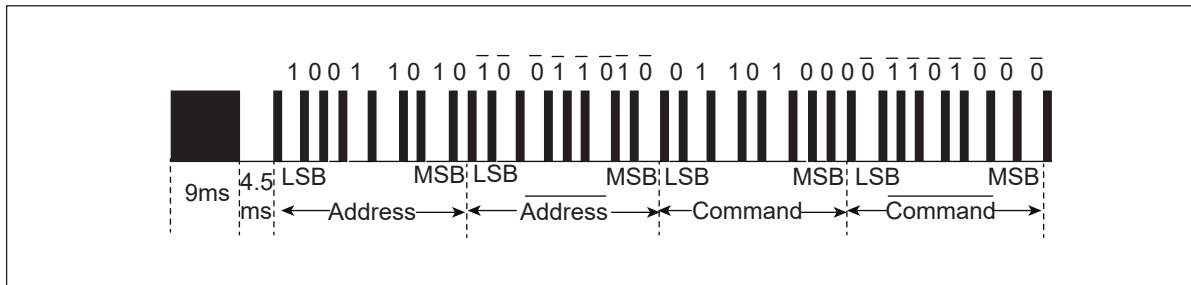


图 14.2: NEC 协议波形

头脉冲是 9ms 的高电平脉冲和 4.5ms 的低电平，之后是 8-bit 的地址码及其反码，然后是 8-bit 的命令码及其反码，尾脉冲是 560us 高电平与 560us 低电平。

- RC-5 协议

RC-5 协议的逻辑 1 与逻辑 0 波形如下图所示：

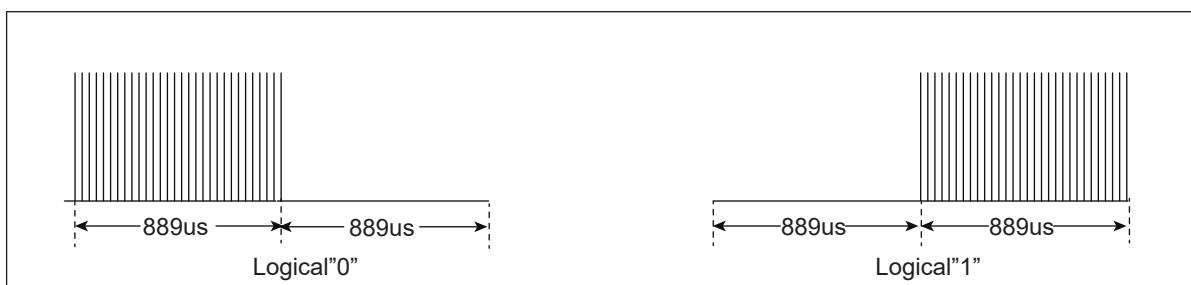


图 14.3: RC5 逻辑波形

逻辑 1 为 1.778ms，先是 889us 的低电平后是 889us 的高电平；逻辑 0 与逻辑 1 波形相反。RC-5 协议的具体格式如下图所示：

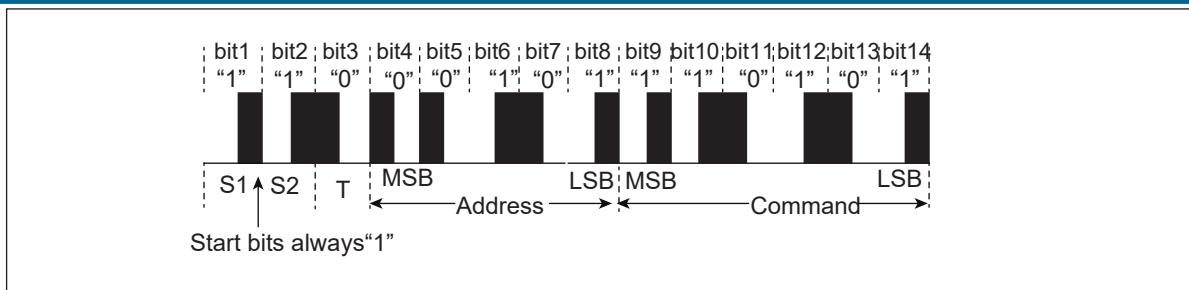


图 14.4: RC5 协议波形

前两位为开始位，固定为逻辑 1，第三位是翻转位，当一个键值发出然后再按下时该位会取反。之后 5 位是地址码与 6 位命令码。需要注意的是，常见的红外一体接收头为了提高接收灵敏度，接收到高电平后输出的是低电平，所以在使用 IR 接收功能时要将接收翻转功能打开。

### 14.3.2 脉冲宽度接收

对于 NEC、RC-5 协议以外的其他任意格式的数据，IR 会以其时钟去依次计数每个高低电平的持续时间，然后将数据存入到深度为 64，宽度为 2 字节的接收 FIFO 之中。

### 14.3.3 普通发送模式

用户可根据具体协议分别对头脉冲、尾脉冲、逻辑 0 和逻辑 1 脉冲进行相应的配置。在设置时需要计算出所使用的协议内各种不同宽度脉冲的公共脉冲宽度单位，即最大公约数，填入寄存器 IRTX\_PULSE\_WIDTH 的低 12 位，各脉冲将其所对应的倍数填入寄存器 IRTX\_PW 中。然后将要发送的逻辑值填入深度为 4，宽度为 4 字节的发送 FIFO 之中。

### 14.3.4 脉冲宽度发送

对于不适用于普通发送模式的协议，IR 提供了脉冲宽度发送的方式。先计算出所使用的协议内各种不同宽度脉冲的公共脉冲宽度单位，即最大公约数，填入寄存器 IRTX\_PULSE\_WIDTH 的低 12 位。然后将从第一个高电平开始到最后一个电平为止的各个电平宽度所对应的倍数（每个倍数占 8-bit）每四个组合成一个 32-bit 数，填入 TX FIFO 中。

### 14.3.5 载波调制

通过设置寄存器 IRTX\_PULSE\_WIDTH 的高 16 位可以产生不同频率和占空比的载波，该寄存器的 <TXMPH1W> 位设置的是载波相位 1 的宽度，<TXMPH0W> 位设置的是载波相位 0 的宽度。

### 14.3.6 自由模式

自由模式下不限制发送的数据总长度，当 TX FIFO 中有数据时就会发送，可配合 DMA 使用。另外，自由模式下的 TX FIFO 宽度可配置，支持 8/16/24/32-bit。

### 14.3.7 DMA 模式

IR TX 支持 DMA 模式，使用该模式需要使能自由模式，并通过寄存器 IRTX\_FIFO\_CONFIG\_1 中 <TFITH> 设置 TX FIFO 的阈值。当该模式启用后，如果 IRTX\_FIFO\_CONFIG\_1 中 <TFICNT> 大于 <TFITH>，则会触发 DMA TX 请求，配置好 DMA 后，DMA 在收到该请求时，会按照设定从内存中将数据搬运到 TX FIFO。

### 14.3.8 IR 中断

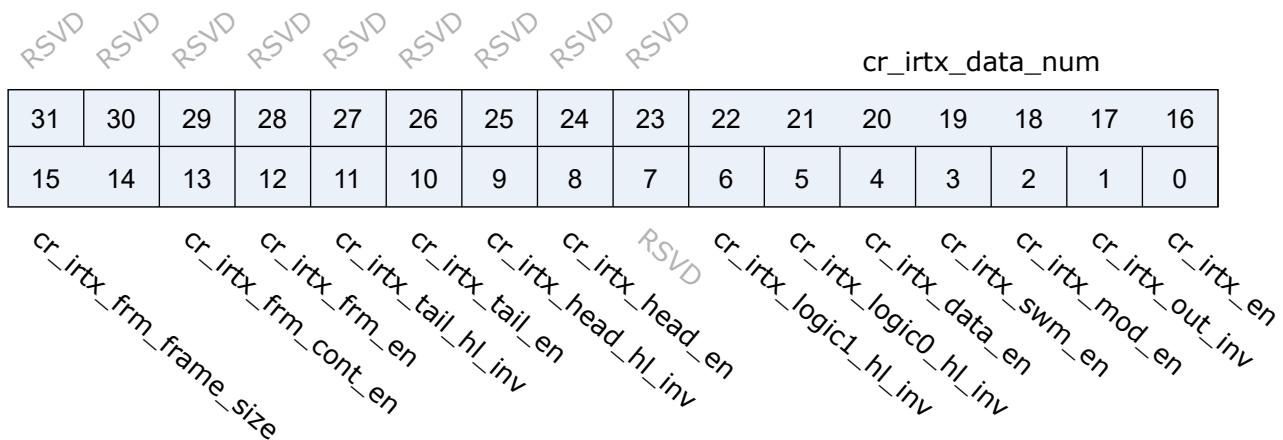
IR 有单独的发送和接收结束中断，当一次发送动作结束时，会产生发送中断。当接收到一段数据时，它会等待电平持续长度达到设定的结束阈值时产生接收中断。可以通过寄存器 IRTX\_INT\_STS 查询发送中断状态和清除中断，通过寄存器 IRRX\_INT\_STS 查询接收中断状态和清除中断。

## 14.4 寄存器描述

名称	描述
irtx_config	
irtx_int_sts	
irtx_pulse_width	
irtx_pw_0	
irtx_pw_1	
irrx_config	
irrx_int_sts	
irrx_pw_config	
irrx_data_count	
irrx_data_word0	
irrx_data_word1	
irtx_fifo_config_0	
irtx_fifo_config_1	
ir_fifo_wdata	
ir_fifo_rdata	

#### 14.4.1 irtx\_config

地址: 0x2000a600



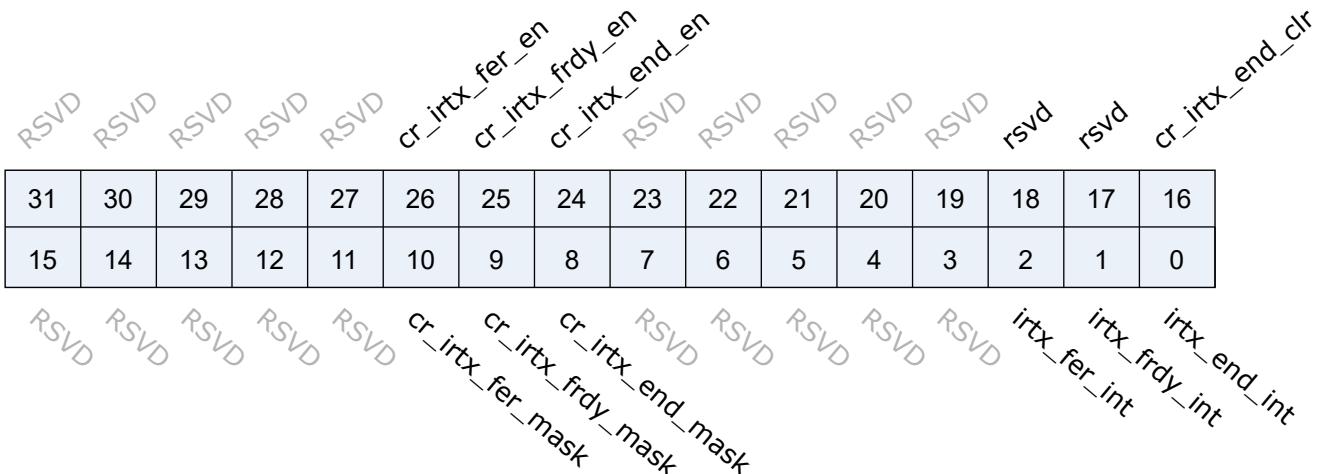
位	名称	权限	复位值	描述
31:23	RSVD			
22:16	cr_irtx_data_num	r/w	7'd31	Bit count of Data phase (unit: bit / PW for normal / SWM) (Don't-care if cr_irtx_frm_en is enabled)
15:14	cr_irtx_frm_frame_size	r/w	2'd0	IRTX freerun mode frame size (also the valid width for each FIFO entry) 2'd0: 8-bit 2'd1: 16-bit 2'd2: 24-bit 2'd3: 32-bit
13	cr_irtx_frm_cont_en	r/w	1'b0	Enable signal of IRTX freerun continuous mode 0: Disabled, each data frame is separated by an idle time (tail_ph0_w+1 + tail_ph1_w+1)*pw_unit 1: Enabled, data continuously sent without interval (if FIFO data is valid)
12	cr_irtx_frm_en	r/w	1'b0	Enable signal of IRTX freerun mode (Don't care if SWM is enabled) Note: HEAD/TAIL is disabled in freerun mode

位	名称	权限	复位值	描述
11	cr_irtx_tail_hl_inv	r/w	1'b0	Tail pulse H/L inverse signal (Don't care if SWM or FRM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
10	cr_irtx_tail_en	r/w	1'b1	Enable signal of tail pulse (Don't care if SWM or FRM is enabled)
9	cr_irtx_head_hl_inv	r/w	1'b0	Tail pulse H/L inverse signal (Don't care if SWM or FRM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
8	cr_irtx_head_en	r/w	1'b1	Enable signal of head pulse (Don't care if SWM or FRM is enabled)
7	RSVD			
6	cr_irtx_logic1_hl_inv	r/w	1'b0	Logic 1 H/L inverse signal (Don't care if SWM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
5	cr_irtx_logic0_hl_inv	r/w	1'b0	Logic 0 H/L inverse signal (Don't care if SWM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
4	cr_irtx_data_en	r/w	1'b1	Enable signal of data phase (Don't care if SWM or FRM is enabled)
3	cr_irtx_swm_en	r/w	1'b0	Enable signal of IRTX Software Mode (SWM)
2	cr_irtx_mod_en	r/w	1'b0	Enable signal of output modulation
1	cr_irtx_out_inv	r/w	1'b0	Output inverse signal 1'b0: Output stays at Low during idle state 1'b1: Output stays at High during idle state

位	名称	权限	复位值	描述
0	cr_irtx_en	r/w	1'b0	Enable signal of IRTX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

#### 14.4.2 irtx\_int\_sts

地址: 0x2000a604



位	名称	权限	复位值	描述
31:27	RSVD			
26	cr_irtx_fer_en	r/w	1'b1	Interrupt enable of irtx_fer_int
25	cr_irtx_frdy_en	r/w	1'b1	Interrupt enable of irtx_frdy_int
24	cr_irtx_end_en	r/w	1'b1	Interrupt enable of irtx_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_irtx_end_clr	w1c	1'b0	Interrupt clear of irtx_end_int
15:11	RSVD			
10	cr_irtx_fer_mask	r/w	1'b1	Interrupt mask of irtx_fer_int
9	cr_irtx_frdy_mask	r/w	1'b1	Interrupt mask of irtx_frdy_int
8	cr_irtx_end_mask	r/w	1'b1	Interrupt mask of irtx_end_int
7:3	RSVD			
2	irtx_fer_int	r	1'b0	IRTX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared

位	名称	权限	复位值	描述
1	irtx_frdy_int	r	1'b1	IRTX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	irtx_end_int	r	1'b0	IRTX transfer end interrupt

#### 14.4.3 irtx\_pulse\_width

地址: 0x2000a610

cr_irtx_mod_ph1_w								cr_irtx_mod_ph0_w							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD RSVD RSVD RSVD cr_irtx_pw_unit															

位	名称	权限	复位值	描述
31:24	cr_irtx_mod_ph1_w	r/w	8'd34	Modulation phase 1 width
23:16	cr_irtx_mod_ph0_w	r/w	8'd17	Modulation phase 0 width
15:12	RSVD			
11:0	cr_irtx_pw_unit	r/w	12'd1124	Pulse width unit

#### 14.4.4 irtx\_pw\_0

地址: 0x2000a614

cr_irtx_logic1_ph1_w								cr_irtx_logic1_ph0_w							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_irtx_logic0_ph1_w cr_irtx_logic0_ph0_w															

位	名称	权限	复位值	描述
31:24	cr_irtx_logic1_ph1_w	r/w	8'd2	Pulse width of logic1 phase 1 (Don't care if SWM is enabled)
23:16	cr_irtx_logic1_ph0_w	r/w	8'd0	Pulse width of logic1 phase 0 (Don't care if SWM is enabled)
15:8	cr_irtx_logic0_ph1_w	r/w	8'd0	Pulse width of logic0 phase 1 (Don't care if SWM is enabled)

位	名称	权限	复位值	描述
7:0	cr_irtx_logic0_ph0_w	r/w	8'd0	Pulse width of logic0 phase 0 (Don't care if SWM is enabled)

#### 14.4.5 irtx\_pw\_1

地址: 0x2000a618

cr_irtx_tail_ph1_w								cr_irtx_tail_ph0_w							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_irtx_head_ph1_w								cr_irtx_head_ph0_w							

位	名称	权限	复位值	描述
31:24	cr_irtx_tail_ph1_w	r/w	8'd0	Pulse width of tail pulse phase 1 (Don't care if SWM is enabled)
23:16	cr_irtx_tail_ph0_w	r/w	8'd0	Pulse width of tail pulse phase 0 (Don't care if SWM is enabled)
15:8	cr_irtx_head_ph1_w	r/w	8'd7	Pulse width of head pulse phase 1 (Don't care if SWM is enabled)
7:0	cr_irtx_head_ph0_w	r/w	8'd15	Pulse width of head pulse phase 0 (Don't care if SWM is enabled)

#### 14.4.6 irrx\_config

地址: 0x2000a640

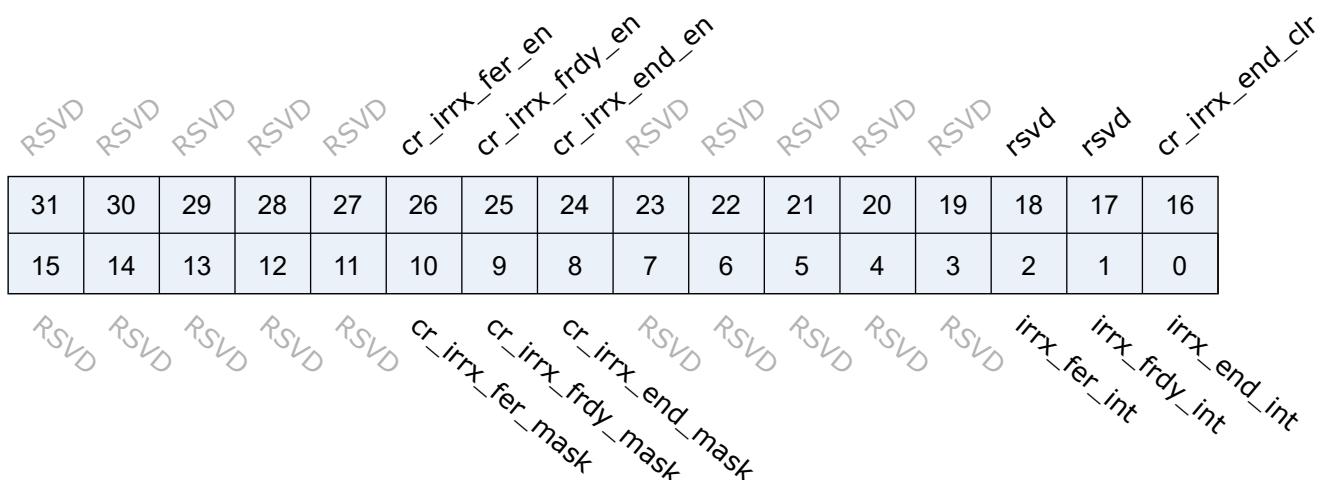
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	cr_irrx_deg_cnt	RSVD	RSVD	RSVD	cr_irrx_mode	cr_irrx_deg_en	cr_irrx_in_inv	cr_irrx_en	cr_irrx_in_inv	cr_irrx_en	cr_irrx_in_inv	cr_irrx_en

位	名称	权限	复位值	描述
31:12	RSVD			

位	名称	权限	复位值	描述
11:8	cr_irrx_deg_cnt	r/w	4'd0	De-glitch function cycle count
7:5	RSVD			
4	cr_irrx_deg_en	r/w	1'b0	Enable signal of IRRX input de-glitch function
3:2	cr_irrx_mode	r/w	2'd0	IRRX mode 0: NEC 1: RC5 2: SW pulse-width detection mode (SWM) 3: Reserved
1	cr_irrx_in_inv	r/w	1'b1	Input inverse signal
0	cr_irrx_en	r/w	1'b0	Enable signal of IRRX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

#### 14.4.7 irrx\_int\_sts

地址: 0x2000a644



位	名称	权限	复位值	描述
31:27	RSVD			
26	cr_irrx_fer_en	r/w	1'b1	Interrupt enable of irrx_fer_int
25	cr_irrx_frdy_en	r/w	1'b1	Interrupt enable of irrx_frdy_int
24	cr_irrx_end_en	r/w	1'b1	Interrupt enable of irrx_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	

位	名称	权限	复位值	描述
16	cr_irrx_end_clr	w1c	1'b0	Interrupt clear of irrx_end_int
15:11	RSVD			
10	cr_irrx_fer_mask	r/w	1'b1	Interrupt mask of irrx_fer_int
9	cr_irrx_frdy_mask	r/w	1'b1	Interrupt mask of irrx_frdy_int
8	cr_irrx_end_mask	r/w	1'b1	Interrupt mask of irrx_end_int
7:3	RSVD			
2	irrx_fer_int	r	1'b0	IRRX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	irrx_frdy_int	r	1'b0	IRRX FIFO ready ( $rx\_fifo\_cnt > rx\_fifo\_th$ ) interrupt, auto-cleared when data is popped
0	irrx_end_int	r	1'b0	IRRX transfer end interrupt

#### 14.4.8 irrx\_pw\_config

地址: 0x2000a648

cr\_irrx\_end\_th

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_irrx\_data\_th

位	名称	权限	复位值	描述
31:16	cr_irrx_end_th	r/w	16'd8999	Pulse width threshold to trigger END condition
15:0	cr_irrx_data_th	r/w	16'd3399	Pulse width threshold for Logic0/1 detection (Don't care if SWM is enabled)

#### 14.4.9 irrx\_data\_count

地址: 0x2000a650

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

sts\_irrx\_data\_cnt

位	名称	权限	复位值	描述
31:7	RSVD			
6:0	sts_irrx_data_cnt	r	7'd0	RX data bit count (pulse-width count for SWM)

#### 14.4.10 irrx\_data\_word0

地址: 0x2000a654

sts\_irrx\_data\_word0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_irrx\_data\_word0

位	名称	权限	复位值	描述
31:0	sts_irrx_data_word0	r	32'h0	RX data word 0

#### 14.4.11 irrx\_data\_word1

地址: 0x2000a658

sts\_irrx\_data\_word1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_irrx\_data\_word1

位	名称	权限	复位值	描述
31:0	sts_irrx_data_word1	r	32'h0	RX data word 1

#### 14.4.12 irtx\_fifo\_config\_0

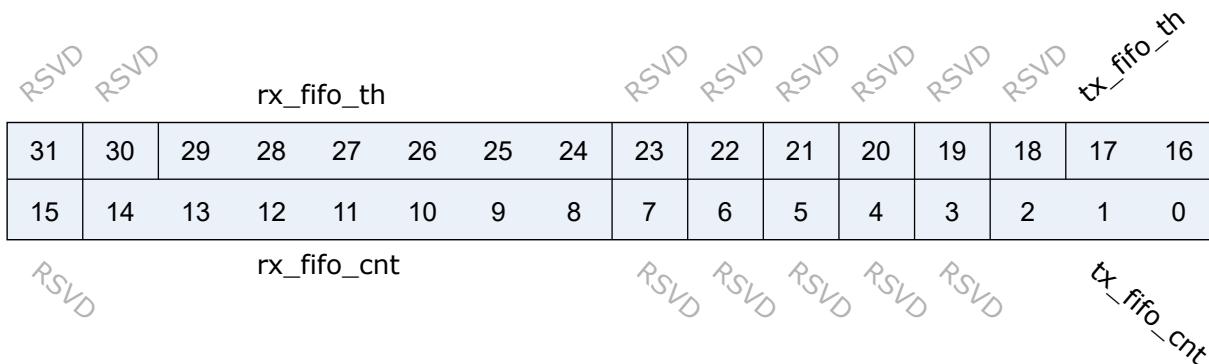
地址: 0x2000a680

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															
rx_fifo_overflow rx_fifo_underflow tx_fifo_overflow tx_fifo_underflow tx_fifo_clr rsvd irtx_dma_en															

位	名称	权限	复位值	描述
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	rsvd	rsvd	1'b0	
0	irtx_dma_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

#### 14.4.13 irtx\_fifo\_config\_1

地址: 0x2000a684



位	名称	权限	复位值	描述
31:30	RSVD			
29:24	rx_fifo_th	r/w	6'd0	RX FIFO threshold, irrx_frdy_int will not be asserted if rx_fifo_cnt is less than this value
23:18	RSVD			
17:16	tx_fifo_th	r/w	2'd0	TX FIFO threshold, irtx_frdy_int & dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15	RSVD			

位	名称	权限	复位值	描述
14:8	rx_fifo_cnt	r	7'd0	RX FIFO available count
7:3	RSVD			
2:0	tx_fifo_cnt	r	3'd4	TX FIFO available count

#### 14.4.14 ir\_fifo\_wdata

地址: 0x2000a688

tx\_fifo\_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tx\_fifo\_wdata

位	名称	权限	复位值	描述
31:0	tx_fifo_wdata	w	32'h0	IRTX FIFO data Normal Mode: Each entry contains a 32-bit data word, LSB is sent first Software Mode: Each entry contains 4 pulse widths, [7:0] is the 1st pulse, [15:8] is the 2nd pulse, etc)

#### 14.4.15 ir\_fifo\_rdata

地址: 0x2000a68c

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

rx\_fifo\_rdata

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	rx_fifo_rdata	r	16'h0	IRRX FIFO pulse width data for Software Mode

## 15.1 简介

串行外设接口（Serial Peripheral Interface Bus, SPI）是一种用于短程通信的同步串行通信接口规范，装置之间使用全双工模式通信，是一个主机和一个或多个从机的主从模式。SPI 使用 4 根线完成全双工的通信，这 4 根信号线分别是：CS（片选）、SCLK（时钟）、MOSI(主机输出从机输入)、MISO(主机输入从机输出)。

## 15.2 主要特征

- 既可作为 SPI 主设备，也可作为 SPI 从设备
- 主从设备都支持 4 种时钟格式（CPOL, CPHA）
- 主从设备都支持 1/2/3/4 字节传输模式
- 发送和接收通道各有深度为 32 个字节的 FIFO
- 自适应的 FIFO 深度变化特性，适配高性能的场景应用
  - 当 Frame 为 32Bits 时，FIFO 的深度为 8
  - 当 Frame 为 24Bits 时，FIFO 的深度为 8
  - 当 Frame 为 16Bits 时，FIFO 的深度为 16
  - 当 Frame 为 8Bits 时，FIFO 的深度为 32
- 可配置 MSB/LSB 传输
- 可调整字节传输顺序
- 灵活的时钟配置，最高可支持 80M 时钟
- 可配置 MSB/LSB 优先传输
- 接收忽略功能，可以设定忽略对指定位置数据的接收
- 支持从设备模式下的超时机制
- 支持 DMA 传输模式

## 15.3 功能描述

### 15.3.1 时钟控制

依照不同的时钟相位以及极性设定，SPI 时钟共有四种模式，可以通过寄存器 `spi_config` 的 `cr_spi_sclk_pol` (CPOL) 和 `cr_spi_sclk_ph` (CPHA) 进行设置。CPOL 用来决定 SCK 时钟信号空闲时的电平，CPOL=0 则空闲电平为低电平，CPOL=1 则空闲电平为高电平。CPHA 用来决定采样时刻，CPHA=0 则在每个周期的第一个时钟沿采样，CPHA=1 则在每个周期的第二个时钟沿采样。通过设置寄存器 `spi_prd_0` 和 `spi_prd_1`，还可以调整时钟的开始和结束电平持续时间、相位 0/1 的时间以及每帧数据之间的间隔。四种模式下的具体设置如下图所示：

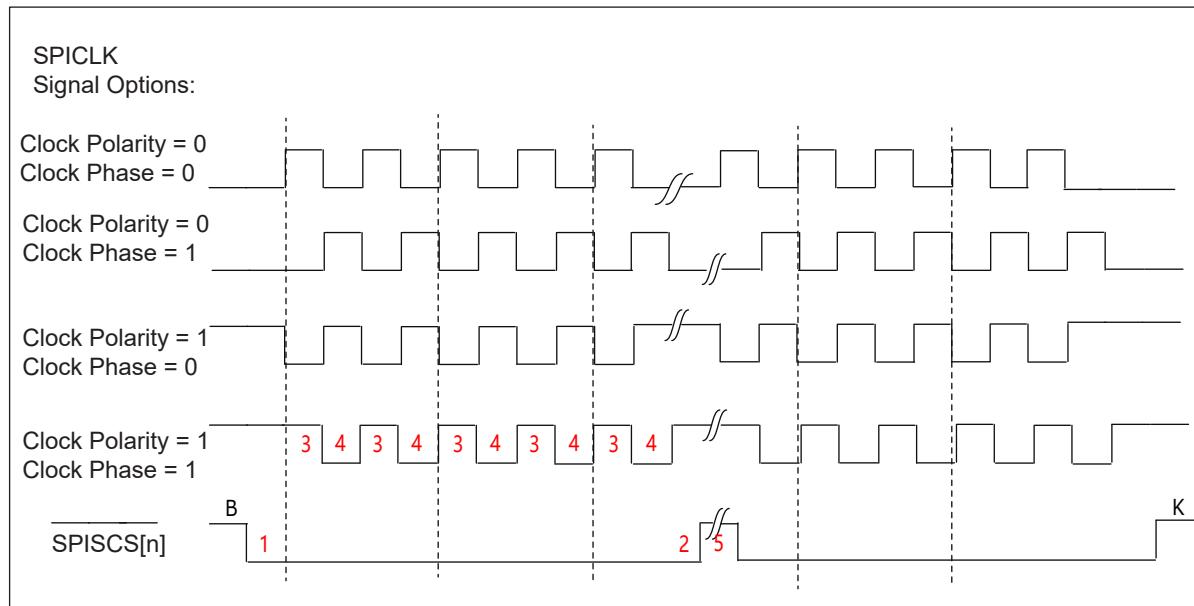


图 15.1: SPI 时序图

其中各数字含义如下：

- 1 是起始条件的长度，通过配置寄存器 `spi_prd_0` 中 `cr_spi_prd_s`。
- 2 是停止条件的长度，通过配置寄存器 `spi_prd_0` 中 `cr_spi_prd_p`。
- 3 是相位 0 的长度，通过配置寄存器 `spi_prd_0` 中 `cr_spi_prd_d_ph_0`。
- 4 是相位 1 的长度，通过配置寄存器 `spi_prd_0` 中 `cr_spi_prd_d_ph_1`。
- 5 是每帧数据之间的间隔，通过配置寄存器 `spi_prd_1` 中 `cr_spi_prd_i`。

### 15.3.2 主设备持续传输模式

开启该模式后，在发送完当前数据而 FIFO 里还存在可用数据时，CS 信号不会被释放。

### 15.3.3 主从设备传输接收数据

主从设备传输接收数据时应设置相同的 framesize，可通过设置寄存器 spi\_config 中 cr\_spi\_frame\_size。如果主设备和从设备的约定以 32bits 的 framesize 进行通信时，在某一帧数据中，主设备的 clk 由于异常不满足 32 时，会出现下列现象。

- 主设备发送的数据无法被送到从设备的 RX FIFO 中。从设备无法接收到数据。
- 从设备发送的数据时，会跳过这一帧数据，等下次主设备 clk 正常时，继续发送下一帧数据。

### 15.3.4 接收忽略功能

通过设置需要过滤掉的开始位和结束位，SPI 会将接收到数据中的对应数据段丢弃。如下图所示：

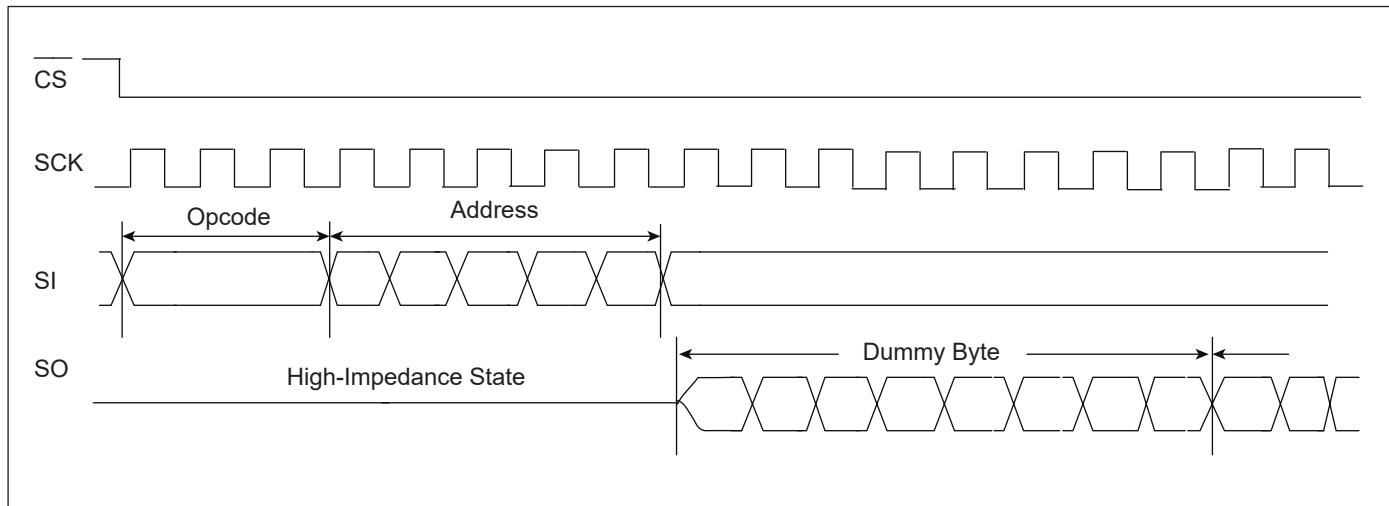


图 15.2: SPI Ignore 波形图

通过配置寄存器 spi\_config 中 cr\_spi\_rxd\_ignr\_en 开启忽略功能。通过配置寄存器 spi\_rxd\_ignr 中 cr\_spi\_rxd\_ignr\_s 设置忽略功能的起始位。通过配置寄存器 spi\_rxd\_ignr 中 cr\_spi\_rxd\_ignr\_p 设置忽略功能的结束位。

上图中过滤的开始位设为 0，结束位设为 7 则 Dummy Byte 会被收到，结束位设为 15 则 Dummy Byte 会被丢弃。

### 15.3.5 滤波功能

通过使能该功能和设置门限值，SPI 会将小于或等于门限值宽度的数据过滤。通过配置寄存器 spi\_config 中 cr\_spi\_deg\_en 使能该功能和配置 cr\_spi\_deg\_cnt 设置门限值，SPI 会将达不到门限值宽度的数据过滤掉。数据宽度小于 cr\_spi\_deg\_cnt+1；如下图所示，若想滤去数据宽度小于 4 的数据，需要将 cr\_urx\_deg\_cnt 的值设置为 4。input 为初始数据，output 为滤波后的数据。

滤波逻辑过程：

- tgl 为 input 和 output 的异或结果。
- deg\_cnt 从 0 开始计数，计数条件为 tgl 为高电平，并且 reached 为低电平。

- 若 deg\_cnt 计数值达到 cr\_urx\_deg\_cnt 设置的值时,reached 为高电平。
- 当 reached 为高电平时, 将 input 输出到 output。
- 注释:deg\_cnt 自加的条件: tgl 为高电平且 reached 为低电平, 其余情况下 deg\_cnt 会被清 0。

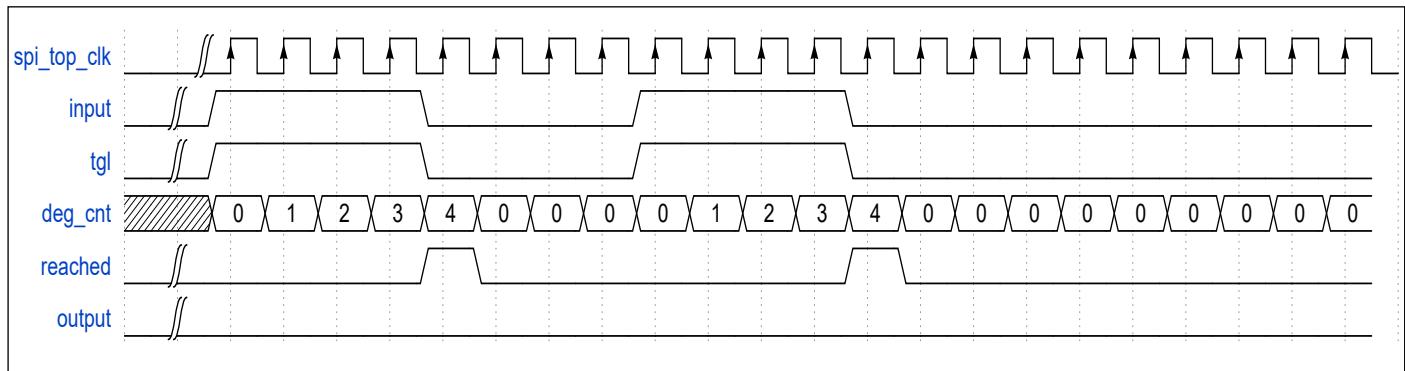


图 15.3: SPI 滤波波形图

### 15.3.6 可配置 MSB/LSB 传输

可配置 MSB/LSB 传输方式仅限于 1 个 byte 中的 8 个 bits 的优先传输顺序, 通过配置寄存器 `spi_config` 中 `cr_spi_bit_inv` 位设置字节内 bit 的传输顺序。0 表示 MSB, 1 表示 LSB。以 frame size 等于 24 bits 传输数据为例, 数据格式为: `Data[23:0]=0x123456`。当设置为 MSB 传输时, 传输的顺序为: 01010110(二进制, 第 1 个字节: 0x56);00110100(二进制, 第 2 个字节: 0x34);00010010(二进制, 第 3 个字节: 0x12); 当设置为 LSB 传输时, 传输的顺序为: 01101010(二进制, 第 1 个字节: 0x56);00101100(二进制, 第 2 个字节: 0x34);01001000(二进制, 第 3 个字节: 0x12)。

### 15.3.7 可调整字节传输顺序

可调整字节传输顺序方式仅限于 FIFO 中不同的 byte 间的优先传输顺序。通过配置寄存器 `spi_config` 中 `cr_spi_byte_inv` 位设置 FIFO 内 byte 的传输顺序。0 表示优先发送低字节, 1 表示优先发送高字节。同样以 frame size 等于 24 bits 传输数据为例, 数据格式为: `Data[23:0]=0x123456`。当设置优先发送低字节, 传输的顺序为: 0x56(第 1 个字节: 低字节); 0x34(第 2 个字节: 中间字节); 0x12(第 3 个字节: 高字节); 当设置优先发送高字节, 传输的顺序为: 0x12(第 3 个字节: 高字节); 0x34(第 2 个字节: 中间字节); 0x56(第 1 个字节: 低字节);

可调整字节传输可以和可配置 MSB/LSB 传输配合使用。

### 15.3.8 从模式超时机制

通过设定一个超时门限，当从模式下 SPI 超过该时间值未收到时钟信号时，会触发中断。

### 15.3.9 I/O 传输模式

芯片通信处理器可以响应来自 FIFO 的中断来执行 FIFO 填充和清空操作。每个 FIFO 都有一个可编程的 FIFO 触发阈值来触发中断。当寄存器 `spi_fifo_config_1` 中 `rx_fifo_cnt` 大于 `rx_fifo_th` 触发阈值时，将产生一个中断，向芯片通信处理器发送信号来清空 RX FIFO。当寄存器 `spi_fifo_config_1` 中 `rx_fifo_cnt` 大于 `rx_fifo_th` 时，将产生中断，向芯片通信处理器发送信号来重新填充 TX FIFO。可以通过查询 SPI 状态寄存器来确定 FIFO 中的采样值以及 FIFO 的状态。软件负责确保正确的 RX FIFO 触发阈值和 TX FIFO 触发阈值，防止接收 FIFO overflow 和发送 FIFO underflow。

### 15.3.10 DMA 传输模式

SPI 支持 DMA 传输模式。使用该模式需要分别设置 TX 和 RX FIFO 的阈值，将寄存器 `spi_fifo_config_0` 中 `spi_dma_tx_en` 置 1，则开启 DMA 发送模式。将寄存器 `spi_fifo_config_0` 中 `spi_dma_rx_en` 置 1，则开启 DMA 接收模式。当该模式启用后，UART 会对 TX/RX FIFO 进行检查，一旦寄存器 `spi_fifo_config_1` 中 `tx_fifo_cnt/rx_fifo_cnt` 大于 `tx_fifo_th/rx_fifo_th`，将会发起 DMA 请求，DMA 会按照设定将数据搬移至 TX FIFO 中或从 RX FIFO 中移出。

### 15.3.11 SPI 中断

SPI 有着丰富的中断控制，包括以下几种中断模式：

- SPI 传输结束中断
- TX FIFO 请求中断
- RX FIFO 请求中断
- 从模式传输超时中断
- 从模式 TX 过载中断
- TX/RX FIFO 溢出中断

在主模式下，SPI 传输结束中断会在每帧数据传输结束时触发；在从模式下，SPI 传输结束中断会在 CS 信号被释放时触发。TX/RX FIFO 请求中断会在其 FIFO 可用计数值大于其设定的阈值时触发，当条件不满足时该中断标志会自动清除。从模式传输超时中断会在从模式下超过超时门限值未收到时钟信号时触发。如果 TX/RX FIFO 发生了上溢或者下溢，会触发 TX/RX FIFO 溢出中断，当 FIFO 清除寄存器 `spi_fifo_config_0` 中位 `tx_fifo_clr/rx_fifo_clr` 被置 1 时，对应的 FIFO 会被清空，同时溢出中断标志会自动清除。可以通过寄存器 `SPI_INT_STS` 查询各中断状态和对相应的位写 1 清除中断。

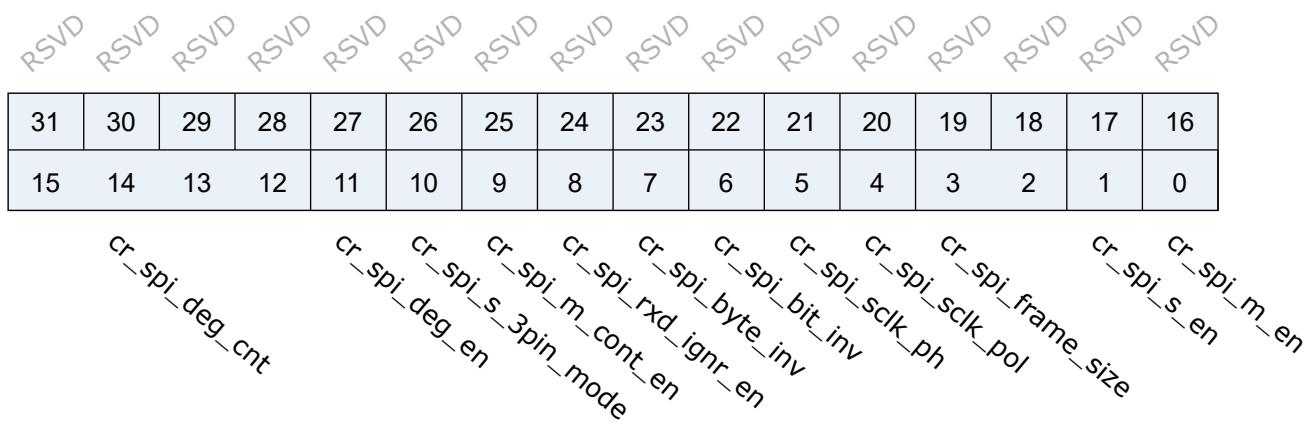
## 15.4 寄存器描述

名称	描述
<code>spi_config</code>	
<code>spi_int_sts</code>	
<code>spi_bus_busy</code>	

名称	描述
spi_prd_0	
spi_prd_1	
spi_rxd_ignr	
spi_sto_value	
spi_fifo_config_0	
spi_fifo_config_1	
spi_fifo_wdata	
spi_fifo_rdata	
backup_io_en	

### 15.4.1 spi\_config

地址: 0x30008000

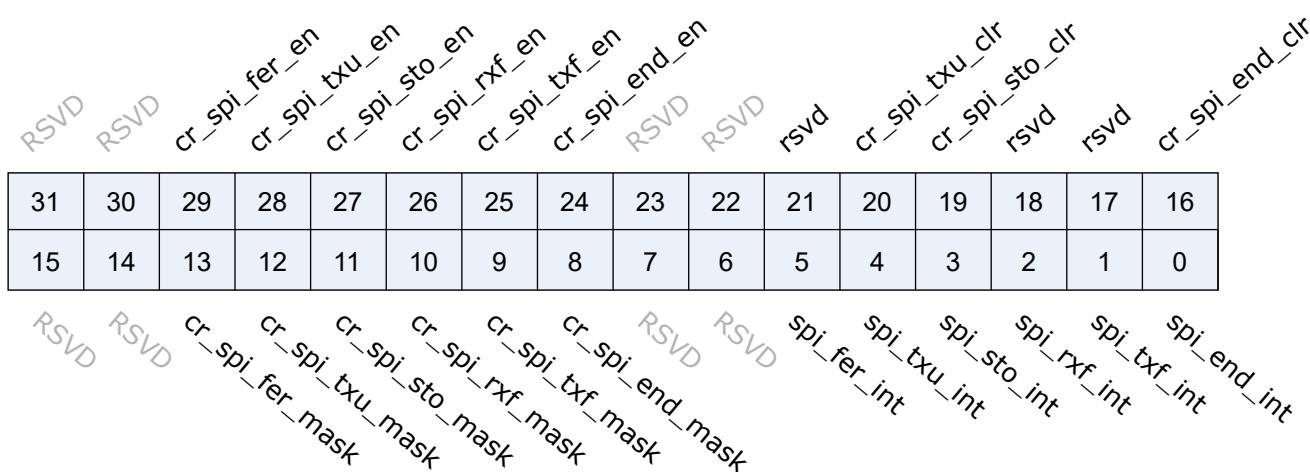


位	名称	权限	复位值	描述
31:16	RSVD			
15:12	cr_spi_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_spi_deg_en	r/w	1'b0	Enable signal of all input de-glitch function
10	cr_spi_s_3pin_mode	r/w	1'b0	SPI slave 3-pin mode 1'b0: 4-pin mode (SS_n is enabled) 1'b1: 3-pin mode (SS_n is disabled / don't care)

位	名称	权限	复位值	描述
9	cr_spi_m_cont_en	r/w	1'b0	<p>Enable signal of master continuous transfer mode</p> <p>1'b0: Disabled, SS_n will de-assert between each data frame</p> <p>1'b1: Enabled, SS_n will stay asserted between each consecutive data frame if the next data is valid in the FIFO</p>
8	cr_spi_rxd_ignr_en	r/w	1'b0	Enable signal of RX data ignore function
7	cr_spi_byte_inv	r/w	1'b0	<p>Byte-inverse signal for each FIFO entry data</p> <p>0: Byte[0] is sent out first</p> <p>1: Byte[3] is sent out first</p>
6	cr_spi_bit_inv	r/w	1'b0	<p>Bit-inverse signal for each data byte</p> <p>0: Each byte is sent out MSB-first</p> <p>1: Each byte is sent out LSB-first</p>
5	cr_spi_sclk_ph	r/w	1'b0	SCLK clock phase inverse signal
4	cr_spi_sclk_pol	r/w	1'b0	<p>SCLK polarity</p> <p>0: SCLK output LOW at IDLE state</p> <p>1: SCLK output HIGH at IDLE state</p>
3:2	cr_spi_frame_size	r/w	2'd0	<p>SPI frame size (also the valid width for each FIFO entry)</p> <p>2'd0: 8-bit</p> <p>2'd1: 16-bit</p> <p>2'd2: 24-bit</p> <p>2'd3: 32-bit</p>
1	cr_spi_s_en	r/w	1'b0	<p>Enable signal of SPI Slave function, Master and Slave should not be both enabled at the same time</p> <p>(This bit becomes don't-care if cr_spi_m_en is enabled)</p>
0	cr_spi_m_en	r/w	1'b0	<p>Enable signal of SPI Master function</p> <p>Asserting this bit will trigger the transaction, and should be de-asserted after finish</p>

### 15.4.2 spi\_int\_sts

地址: 0x30008004



位	名称	权限	复位值	描述
31:30	RSVD			
29	cr_spi_fer_en	r/w	1'b1	Interrupt enable of spi_fer_int
28	cr_spi_txu_en	r/w	1'b1	Interrupt enable of spi_txu_int
27	cr_spi_sto_en	r/w	1'b1	Interrupt enable of spi_sto_int
26	cr_spi_rxf_en	r/w	1'b1	Interrupt enable of spi_rxf_int
25	cr_spi_txf_en	r/w	1'b1	Interrupt enable of spi_txf_int
24	cr_spi_end_en	r/w	1'b1	Interrupt enable of spi_end_int
23:22	RSVD			
21	rsvd	rsvd	1'b0	
20	cr_spi_txu_clr	w1c	1'b0	Interrupt clear of spi_txu_int
19	cr_spi_sto_clr	w1c	1'b0	Interrupt clear of spi_sto_int
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_spi_end_clr	w1c	1'b0	Interrupt clear of spi_end_int
15:14	RSVD			
13	cr_spi_fer_mask	r/w	1'b1	Interrupt mask of spi_fer_int
12	cr_spi_txu_mask	r/w	1'b1	Interrupt mask of spi_txu_int
11	cr_spi_sto_mask	r/w	1'b1	Interrupt mask of spi_sto_int
10	cr_spi_rxf_mask	r/w	1'b1	Interrupt mask of spi_rxf_int

位	名称	权限	复位值	描述
9	cr_spi_txf_mask	r/w	1'b1	Interrupt mask of spi_txe_int
8	cr_spi_end_mask	r/w	1'b1	Interrupt mask of spi_end_int
7:6	RSVD			
5	spi_fer_int	r	1'b0	SPI TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
4	spi_txu_int	r	1'b0	SPI slave mode TX underrun error flag, triggered when TxD is not ready during transfer in slave mode
3	spi_sto_int	r	1'b0	SPI slave mode transfer time-out interrupt, triggered when SPI bus is idle for a given value
2	spi_rxf_int	r	1'b0	SPI RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
1	spi_txf_int	r	1'b1	SPI TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	spi_end_int	r	1'b0	SPI transfer end interrupt, shared by both master and slave mode  Master mode: Triggered when the final frame is transferred  Slave mode: Triggered when CS_n is de-asserted

### 15.4.3 spi\_bus\_busy

地址: 0x30008008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	sts_spi_bus_busy

位	名称	权限	复位值	描述
31:1	RSVD			
0	sts_spi_bus_busy	r	1'b0	Indicator of SPI bus busy

#### 15.4.4 spi\_prd\_0

地址: 0x30008010

cr\_spi\_prd\_d\_ph\_1

cr\_spi\_prd\_d\_ph\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_spi\_prd\_p

cr\_spi\_prd\_s

位	名称	权限	复位值	描述
31:24	cr_spi_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 (please refer to "Timing" tab)
23:16	cr_spi_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (please refer to "Timing" tab)
15:8	cr_spi_prd_p	r/w	8'd15	Length of STOP condition (please refer to "Timing" tab)
7:0	cr_spi_prd_s	r/w	8'd15	Length of START condition (please refer to "Timing" tab)

#### 15.4.5 spi\_prd\_1

地址: 0x30008014

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_spi\_prd\_i

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	cr_spi_prd_i	r/w	8'd15	Length of INTERVAL between frame (please refer to "Timing" tab)

### 15.4.6 spi\_rxd\_ignr

地址: 0x30008018

cr_spi_rxd_ignr_s															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_spi_rxd_ignr_p															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

位	名称	权限	复位值	描述
31:21	RSVD			
20:16	cr_spi_rxd_ignr_s	r/w	5'd0	Starting point of RX data ignore function
15:5	RSVD			
4:0	cr_spi_rxd_ignr_p	r/w	5'd0	Stopping point of RX data ignore function

### 15.4.7 spi\_sto\_value

地址: 0x3000801c

cr_spi_sto_value															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:12	RSVD			
11:0	cr_spi_sto_value	r/w	12'hFFF	Time-out value for spi_sto_int triggering

### 15.4.8 spi\_fifo\_config\_0

地址: 0x30008080

位	名称	权限	复位值	描述
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	spi_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	spi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

## 15.4.9 spi\_fifo\_config\_1

地址: 0x30008084

位	名称	权限	复位值	描述
31:29	RSVD			
28:24	rx_fifo_th	r/w	5'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:21	RSVD			
20:16	tx_fifo_th	r/w	5'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:14	RSVD			
13:8	rx_fifo_cnt	r	6'd0	RX FIFO available count (unit: byte)
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count (unit: byte)

#### 15.4.10 spi\_fifo\_wdata

地址: 0x30008088

spi\_fifo\_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

spi\_fifo\_wdata

位	名称	权限	复位值	描述
31:0	spi_fifo_wdata	w	x	<p>TX FIFO write data port</p> <p>Note: Partial valid if cr_spi_frame_size is set to different value:</p> <ul style="list-style-type: none"> <li>2'd0 (8-bit frame): Only [7:0] are valid and [31:8] are don't-care</li> <li>2'd1 (16-bit frame): Only [15:0] are valid and [31:16] are don't-care</li> <li>2'd2 (24-bit frame): Only [23:0] are valid and [31:24] are don't-care</li> <li>2'd3 (32-bit frame): Entire [31:0] are valid</li> </ul>

## 15.4.11 spi\_fifo\_rdata

地址: 0x3000808c

spi\_fifo\_rdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

spi\_fifo\_rdata

位	名称	权限	复位值	描述
31:0	spi_fifo_rdata	r	32'h0	<p>RX FIFO read data port</p> <p>Note: Partial valid if cr_spi_frame_size is set to different value:</p> <ul style="list-style-type: none"> <li>2'd0 (8-bit frame): Only [7:0] are valid and [31:8] are all 0s</li> <li>2'd1 (16-bit frame): Only [15:0] are valid and [31:16] are all 0s</li> <li>2'd2 (24-bit frame): Only [23:0] are valid and [31:24] are all 0s</li> <li>2'd3 (32-bit frame): Entire [31:0] is valid</li> </ul>

## 15.4.12 backup\_io\_en

地址: 0x300080fc

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

位	名称	权限	复位值	描述
31:1	RSVD			
0	backup_io_en	r/w	1'b0	

# 16

## UART

### 16.1 简介

通用异步收发传输器（Universal Asynchronous Receiver/Transmitter，通常称为 UART）是一种异步收发传输器，提供了与外部设备进行全双工数据交换的灵活方式。BL808 共有 4 组 UART，配合 DMA 使用，可以实现高效的数据通信。

### 16.2 主要特征

- 全双工异步通信
- 数据位长度可选择 5/6/7/8 比特
- 停止位长度可选择 0.5/1/1.5/2 比特
- 支持奇/偶/无校验比特
- 可侦测错误的起始比特
- 丰富的中断控制
- 支持硬件流控（RTS/CTS）
- 便捷的波特率编程
- 可配置 MSB/LSB 优先传输
- 普通/固定字符的自动波特率检测
- 32 字节发送/接收 FIFO
- 支持 DMA 传输模式
- 支持 10Mbps 及其以上波特率
- 支持 LIN 总线协议
- 支持 RS485 模式
- 时钟源可以选择 160M/BCLK/XCLK
- 支持滤波功能

## 16.3 功能描述

### 16.3.1 数据格式描述

正常的 UART 通信数据是由起始位、数据位、奇偶校验位、停止位组成的。BL808 的 UART 支持可配置的数据位、奇偶校验位和停止位，这些都在寄存器 `utx_config` 和 `urx_config` 中设置。一帧数据的波形如下图所示：

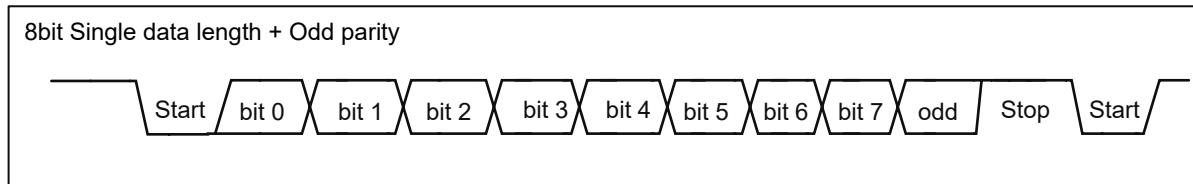


图 16.1: UART 数据格式

数据帧的起始位占用 1-bit，停止位可以通过配置寄存器 `utx_config` 中位 `cr_utx_bit_cnt_p` 实现 0.5/1/1.5/2 位宽。起始位为低电平，停止位为高电平。数据位宽可以通过寄存器 `utx_config` 中位 `cr_utx_bit_cnt_d` 配置为 5/6/7/8 位宽。当置位寄存器 `utx_config` 中位 `cr_utx_prt_en` 和寄存器 `urx_config` 中位 `cr_urx_prt_en` 时，数据帧会在数据之后添加一位奇偶校验位。寄存器 `utx_config` 中位 `cr_utx_prt_sel` 和寄存器 `urx_config` 中位 `cr_urx_prt_sel` 用于选择奇校验还是偶校验。当接收器检测到输入数据的校验位错误时会产生校验错误中断。但是接收的数据仍然会进入 FIFO。奇校验的计算方法：如果当前数据位 1 的个数是奇数个，奇校验位为 0；反之为 1。偶校验的计算方法：如果当前数据位 1 的个数是奇数个，偶校验位为 1；反之为 0。

### 16.3.2 基本架构图

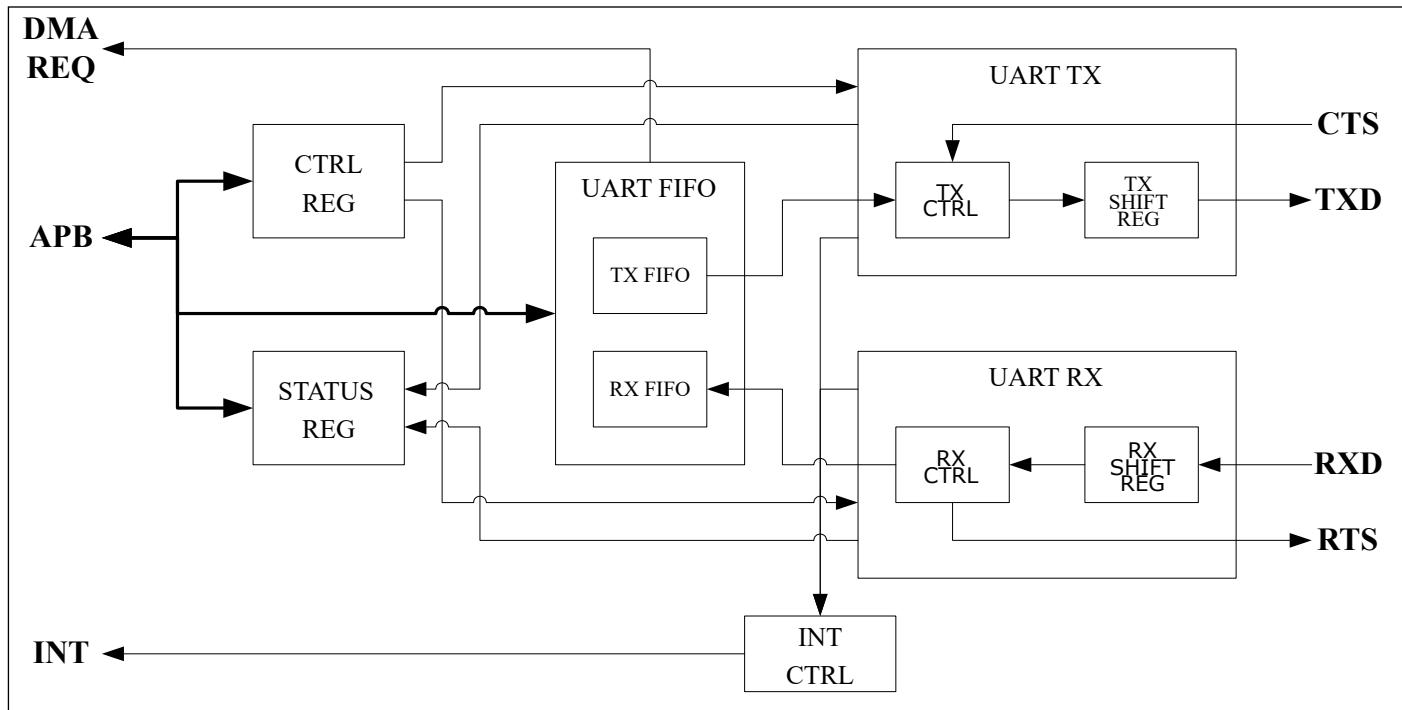


图 16.2: UART 架构图

### 16.3.3 时钟源

UART 有 3 个时钟源：XCLK, 160MHz CLK 以及 BCLK。时钟中的分频器用于对时钟源进行分频，然后产生时钟信号来驱动 UART 模块。如下图所示：

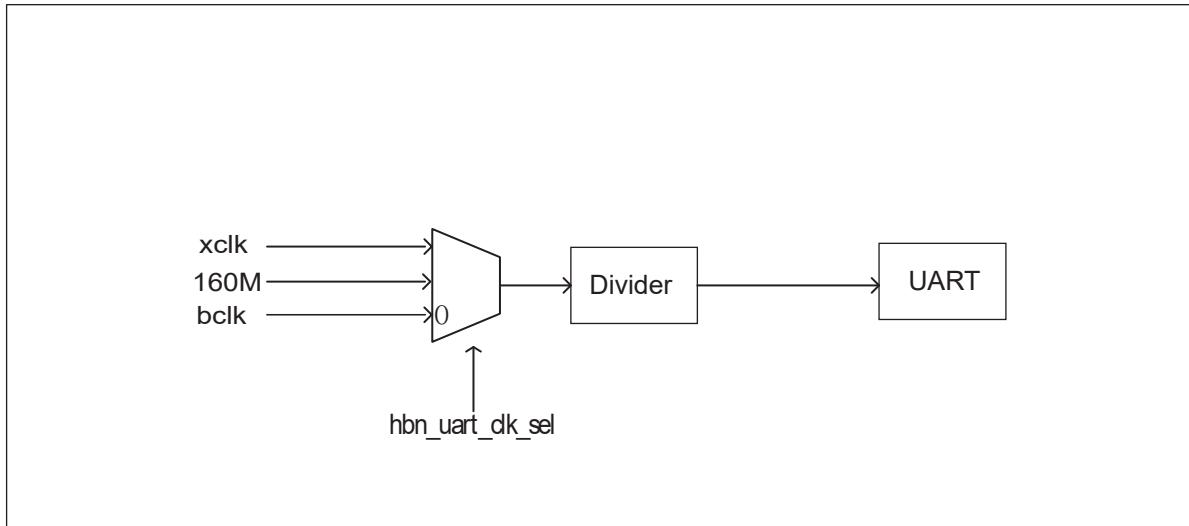


图 16.3: UART 时钟

### 16.3.4 波特率设定

用户可通过设置寄存器 `UART_BIT_PRD` 来产生所需的波特率，该寄存器的高 16 位 `cr_urx_bit_prd` 和低 16 位 `cr_utx_bit_prd` 分别对应 RX 与 TX，即 RX 与 TX 的波特率可单独进行设置，该 16 位值需要通过计算得出，公式如下： $\text{波特率} = \text{UART 时钟} / (16 \text{ 位宽系数} + 1)$  即： $16 \text{ 位位宽系数} = \text{UART 时钟}/\text{波特率}-1$  该 16 位位宽系数的含义是以 UART 时钟去计数当前波特率位宽所得到的计数值。由于 16 位位宽系数最大值为 65535，所以 UART 支持的最小波特率为： $\text{UART 时钟}/65536$ 。在 UART 对数据进行采样之前，会先对数据进行滤波，将波形当中的毛刺滤掉。然后会在上述 16 位系数的中间值时刻进行采样，这样根据不同的波特率调整不同的采样时刻，以保持其采到的始终是中间值，大大提高了灵活性与精度。采样过程如下图所示：

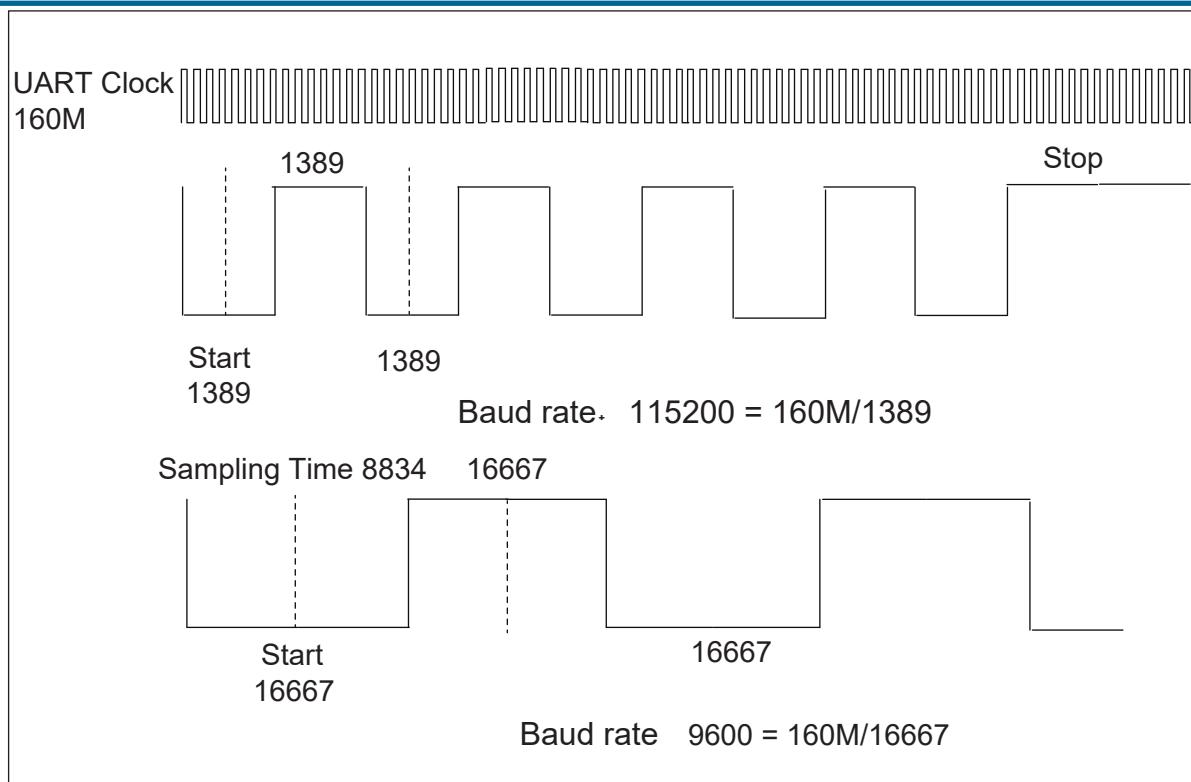


图 16.4: UART 采样波形图

### 16.3.5 滤波

通过配置寄存器 `urx_config` 中 `cr_urx_deg_en` 使能该功能和配置 `cr_urx_deg_cnt` 设置门限值，UART 会将达不到门限值宽度的数据过滤掉如下图所示，若想滤去数据宽度小于 4 的数据，需要将 `cr_urx_deg_cnt` 的值设置为 4。`input` 为初始数据，`output` 为滤波后的数据。

滤波逻辑过程：

- `tgl` 为 `input` 和 `output` 的异或结果。
- `deg_cnt` 从 0 开始计数，计数条件为 `tgl` 为高电平，并且 `reached` 为低电平。
- 若 `deg_cnt` 计数值达到 `cr_urx_deg_cnt` 设置的值时，`reached` 为高电平。
- 当 `reached` 为高电平时，将 `input` 输出到 `output`。
- 注释: `deg_cnt` 自加的条件：`tgl` 为高电平且 `reached` 为低电平，其余情况下 `deg_cnt` 会被清 0。

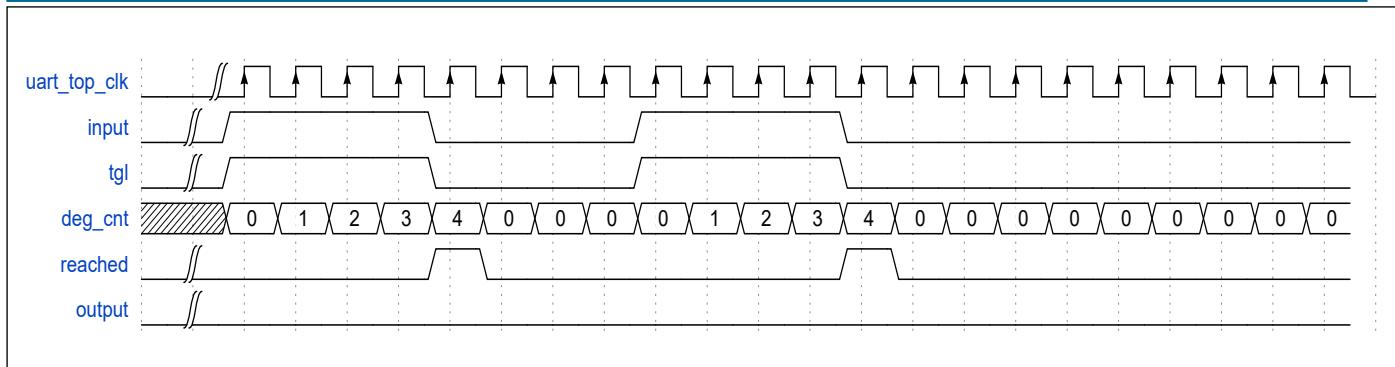


图 16.5: UART 滤波波形图

### 16.3.6 发送器

发送器包含一个 32 字节的发送 FIFO，用来存放待发送的数据。软件可以通过 APB 总线写 TX FIFO，也可以通过 DMA 将数据搬入 TX FIFO。当发送使能位被设置时，FIFO 中存放的数据会从 TX 引脚输出。软件可以通过寄存器 `uart_fifo_config_1` 中 `tx_fifo_cnt` 查询 TX FIFO 剩余可用空间计数值来检查发送器的状态。

发送器的自由运行（FreeRun）模式如下：

- 如果没有开启自由运行（FreeRun）模式，则当发送字节达到指定长度时发送行为终止并产生中断，如果要继续发送则需重新关闭再使能发送使能位。
- 如果开启自由运行（FreeRun）模式，则当 TX FIFO 里存在数据时，发送器就会进行发送，不会因为发送字节达到指定长度而终止。

### 16.3.7 接收器

接收器包含一个 32 字节的接收 FIFO，用来存放接收到的数据。软件可以通过寄存器 `uart_fifo_config_1` 中 `rx_fifo_cnt` 查询 RX FIFO 可用数据计数值来检查接收器的状态。寄存器 `URX_RTO_TIMER` 的低 8 位用于设定一个接收超时门限，当接收器超过该时间值未收到数据时，会触发中断。寄存器 `urx_config` 中 `cr_urx_deg_en` 和 `cr_urx_deg_cnt` 用于使能去毛刺功能和设置门限值，其控制的是 UART 采样之前的滤波部分，UART 会将波形当中宽度低于门限值的毛刺滤掉，然后在将其送去采样。

### 16.3.8 自动波特率检测

UART 模块支持自动波特率检测，该检测分为两种，一种是通用模式，一种是固定字符模式。置位寄存器 `urx_config` 中 `cr_urx_abr_en` 每次开启时，这两种检测模式都会启用。

#### 通用模式

对于所接收到的任意字符数据，UART 模块会计数起使位宽当中的时钟数，此数字会接着被写入寄存器 `STS_URX_ABR_PRD` 的低 16 位 `sts_urx_abr_prd_start` 并用以计算波特率，因此当最先接收到的数据位为 1 时即可得到正确的波特率，如 `LSB-FIRST` 下的'0x01'。

#### 固定字符模式

该模式下，UART 模块在计数起使位宽当中的时钟数后，会继续计数后续数据位的时钟数，并与起始位相比较，如果上下浮动在允许误差范围内，则通过检测，否则计数值会被丢弃。允许误差可以通过设置寄存器 `urx_abr_pw_tol` 中的 `cr_urx_abr_pw_tol` 位来设置，单位是 UART 的时钟源。因此，只有在 `LSB-FIRST` 下接收到固定字符'0x55'/'0xD5' 或 `MSB-FIRST` 下的'0xAA'/'0xAB' 时，UART

模块才会将起使位宽当中的时钟数计数值写入寄存器 `STS_URX_ABR_PRD` 的高 16 位 `sts_urx_abr_prd_0x55`。如下图所示：

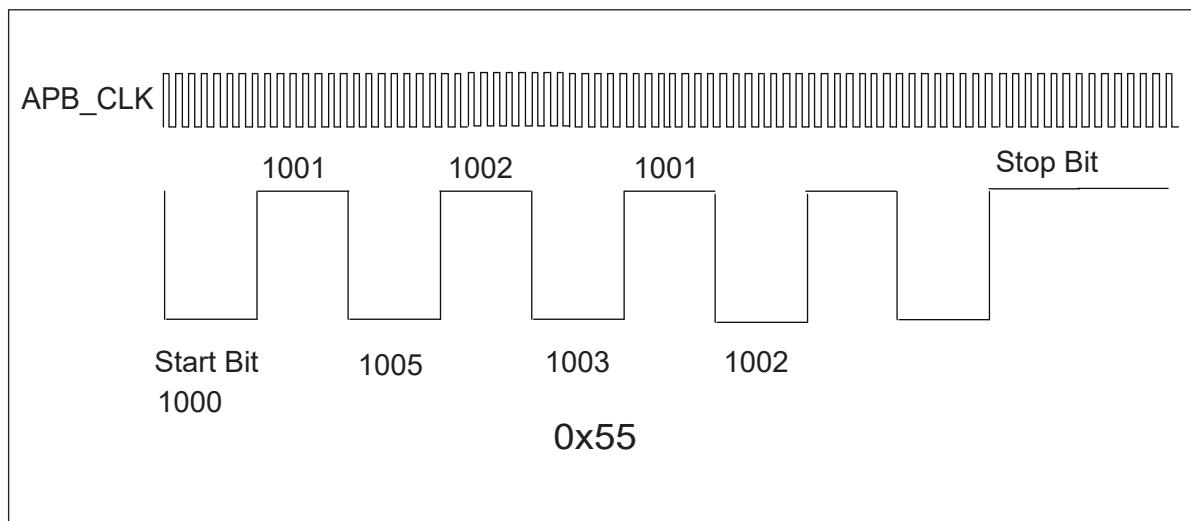


图 16.6: UART 固定字符模式波形图

对于某一未知的波特率，UART 用 `UART_CLK` 去计数起始位的位宽为 1000，第二位的位宽为 1001，与前一位宽上下浮动不超过 4 个 `UART_CLK`，则 UART 会继续计数第三位，第三位为 1005，与起始位相差超过 4，则检测不通过，数据丢弃。UART 会依次将数据位的前 6 位位宽与起始位进行比较。

计算检测到的波特率的公式如下：波特率 = 源时钟 / (16 位检测值 + 1)

### 16.3.9 硬件流控

UART 支持 CTS/RTS 方式的硬件流控，以防止 FIFO 里的数据由于来不及处理而丢失。硬件流控连接如下图所示：

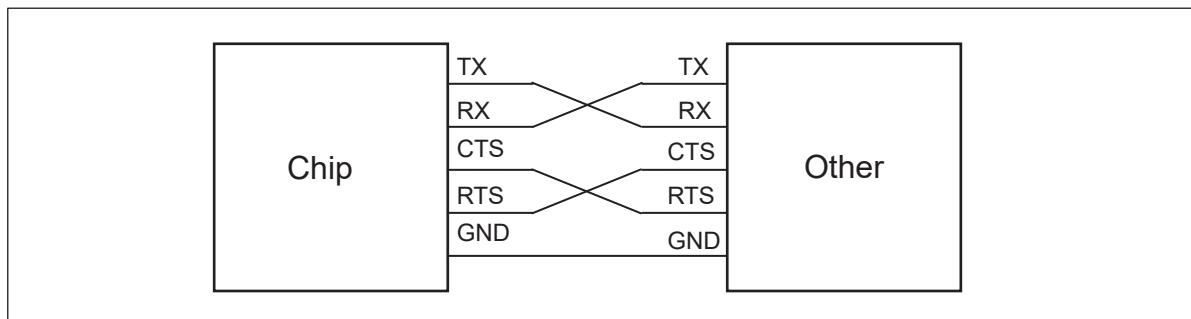


图 16.7: UART 硬件流控图

**RTS** (Require To Send, 发送请求) 为输出信号，用于指示芯片准备好可接收对方数据，低电平有效，低电平说明芯片可以接收数据。

**CTS** (Clear To Send, 清除发送) 为输入信号，用于判断芯片是否可以向对方发送数据，低电平有效，低电平说明芯片可以向对方发送数据。

当使用硬件流控功能时，芯片端输出信号 **RTS** 为低电平表示请求对方发送数据，**RTS** 为高电平表示通知对方中止数据发送。当芯片检测到输入信号 **CTS** 拉高时，**TX** 会停止发送数据，直到检测到 **CTS** 拉低时再继续发送。**CTS** 在通信过程中的任意时刻拉高或

者拉低，不影响 TX 发送数据的连续性，对方收到的数据也是连续的。

发送器的硬件流控有两种方式：

- 寄存器 `uart_sw_mode` 中 `cr_urx_rts_sw_mode` 等于 0：寄存器 `urx_config` 中 `cr_urx_en` 没有开或者是 RX FIFO 快要满（剩一个字节）的时候 RTS 拉高。
- 寄存器 `uart_sw_mode` 中 `cr_urx_rts_sw_mode` 等于 1：可以通过配置寄存器 `URX_SW_MODE` 中 `cr_urx_rts_sw_val` 改变 RTS 的电平。

### 16.3.10 DMA 传输模式

UART 支持 DMA 传输模式。使用该模式需要通过寄存器 `uart_fifo_config_1` 中 `tx_fifo_th` 和 `rx_fifo_th` 分别设置 TX 和 RX FIFO 的阈值，当该模式启用后，如果 `uart_fifo_config_1` 中 `tx_fifo_cnt` 大于 `tx_fifo_th`，则会触发 DMA TX 请求，配置好 DMA 后，DMA 在收到该请求时，会按照设定从内存中将数据搬运到 TX FIFO。如果 `uart_fifo_config_1` 中 `rx_fifo_cnt` 大于 `rx_fifo_th`，则会触发 DMA RX 请求，配置好 DMA 后，DMA 在收到该请求时，会按照设定将 RX FIFO 的数据搬运到内存。传输模式下为了保证芯片 DMA TX Channel 搬运数据的正确性，Channel 配置中需要满足以下条件：`transferWidth` 乘以 `burstSize` 小于等于设置的 `tx_fifo_th` 加 1。传输模式下为了保证芯片 DMA RX Channel 搬运数据的完整性，Channel 配置中需要满足以下条件：`transferWidth` 乘以 `burstSize` 等于设置的 `tx_fifo_th` 加 1。

### 16.3.11 LIN 总线支持

本地互联网络 (LIN) 协议基于 Volvo 衍生公司 Volcano 通信技术公司 (VCT) 开发的 Volcano-Lite 技术。LIN 是 CAN 和 SAE J1850 协议的补充性协议，针对时间要求不高或不需要精确容错的应用（因为 LIN 没有 CAN 协议那样可靠）。LIN 的目标是易于使用，作为 CAN 协议的低成本替代品。LIN 在车辆中可以使用的场合包括车窗升降器、后视镜、雨刷和雨量传感器。

UART 模块支持 LIN 总线模式，一个典型的 LIN 数据传输如图所示。

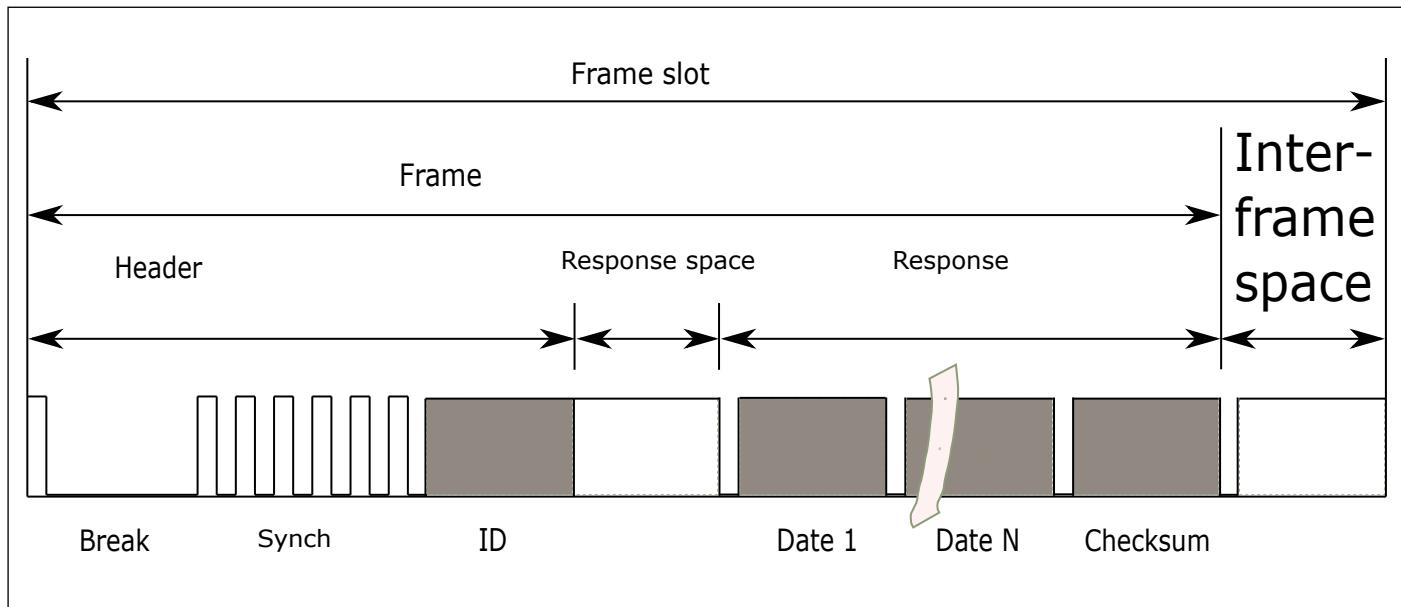


图 16.8: 一个典型的 LIN 帧

LIN 总线采用主从模式，数据总是由主节点发起，主节点发送的帧（头）包含三个部分：同步间隔字段、同步字节字段和一个标识符字段。

同步间隔字段表示报文的开始，至少 13 个显性位（包括起始位）。同步间隔以一个“间隔分隔符”结束，该分隔符至少包含一个隐性位。

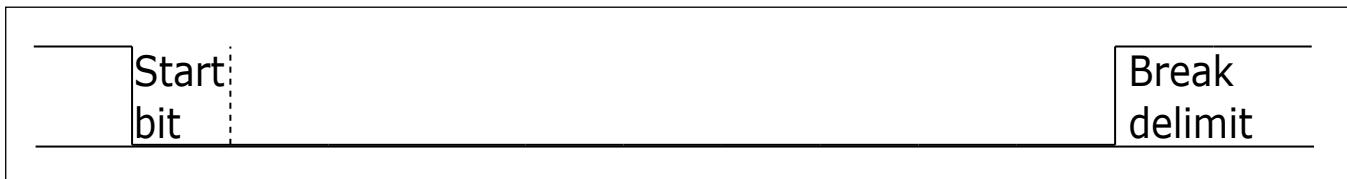


图 16.9: LIN 的 Break 域

LIN 帧中的 Break 长度可以通过 utx\_config 中 cr\_utx\_bit\_cnt\_b 来设定。

发送同步字节字段来确定两个下降沿之间的时间，从而确定主节点使用的传输速率。位模式是 0x55 (01010101，最大下降沿数量)。

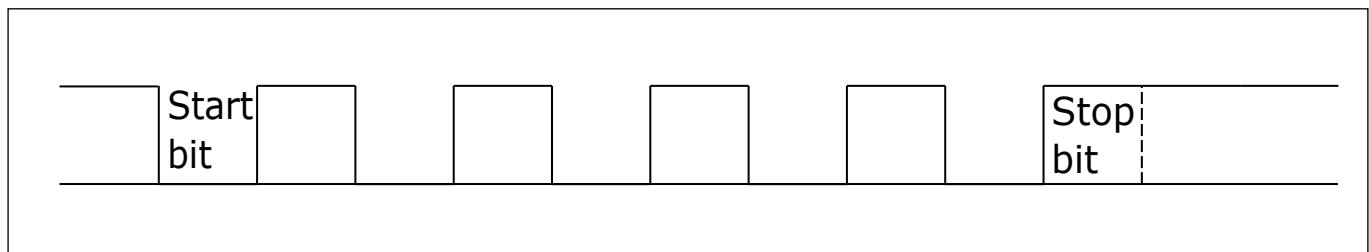


图 16.10: LIN 的 Sync 域

标识符字段包含 6 位长的标识符和两个奇偶校验位。6 位标识符包含关于发送方和接收方的信息，以及响应中要求的字节数。奇偶校验位如下进行计算：校验位 P0 是 ID0、ID1、ID2 和 ID4 之间进行逻辑“或”运算的结果。校验位 P1 是 ID1、ID3、ID4 和 ID5 之间逻辑“或”运算后在进行反转的结果。

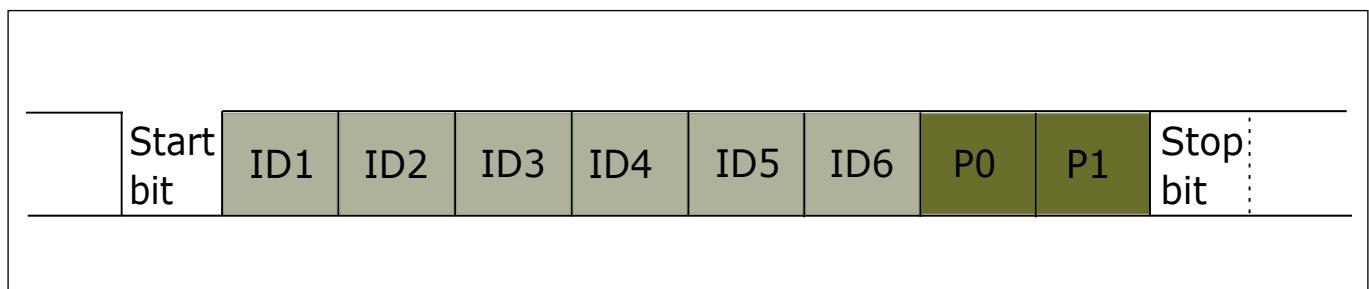


图 16.11: LIN 的 ID 域

从节点等待同步间隔字段，然后通过同步字节字段开始主从节点之间的同步。根据主节点发送的标识符，从节点将进行接收、发送或什么都不做。应该进行发送的从节点发送主节点请求的字节数，然后以一个检验和字段结束传输。

UART 支持 LIN 传输模式。使用该模式需要通过设置寄存器 utx\_config 中 cr\_utx\_lin\_en，通过配置 cr\_utx\_bit\_cnt\_b，使得同步间隔段至少是由 13 位的显性电平组成。

### 16.3.12 RS485 模式

UART 模块支持 485 模式，通过设置寄存器 UTX\_RS485\_CFG 中 cr\_utx\_rs485\_en 可以让 UART 模块工作在 485 模式下，此时，可以通过外接一个 485 的收发器连接到 485 总线上，在该模式下，模块中的 RTS 引脚，用作 485 收发器的 Dir 功能，当 UART 模块有数据需要发生时，会自动控制 RTS 引脚为高电平，让 485 收发器把数据发送到总线上，反之，当 UART 模块没有数据需要发送时，会自动控制 RTS 为低电平，让 485 收发器处在接收状态。

UART 支持 RS485 传输模式。使用该模式需要通过设置寄存器 UTX\_RS485\_CFG 中 cr\_utx\_rs485\_pol 和 cr\_utx\_rs485\_en。

### 16.3.13 UART 中断

UART 有着丰富的中断控制，包括以下几种中断模式：

- TX 传输结束中断
- RX 接收结束中断
- TX FIFO 请求中断
- RX FIFO 请求中断
- RX 超时中断
- RX 奇偶校验错误中断
- TX FIFO 溢出中断
- RX FIFO 溢出中断
- RX BCR 中断
- LIN 同步错误中断
- 通用模式自动波特率检测中断
- 固定字符模式自动波特率检测中断

### 16.3.14 TX/RX 传输结束中断

TX 和 RX 可以通过寄存器 utx\_config 和 urx\_config 的高 16 位分别设置一个传输长度值，当传输的字节数达到这个数值时，就会触发对应的 TX/RX 传输结束中断。在产生该中断的同时，TX 功能停止，用户如果需要继续使用 TX，必须重新初始化该功能模块。RX 功能可以正常使用。对于 TX 而言，如果设置的传输长度值小于 TX 实际发送的数据时，对方可以收到等于 TX 传输长度值的数据，多余的数据存在 TX 的 FIFO 中，新初始化该功能模块后，可以将 TX FIFO 中的数据发出。

例如设置 TX/RX 传输长度值为 64，芯片先发送 63 字节，无中断产生，再次发送 1 字节，此时发送的长度达到 TX 设置的传输长度值时产生发送结束中断，产生中断后再次发送数据发送不出去，数据存储在 TX FIFO 中。对方先发送 63 字节数给芯片，无中断产生，再次发送 1 个字节，此时芯片接收的长度达到 RX 设置的传输长度值时产生中断，再次发送 64 个字节，UART 可以收到数据，FIFO 数据不为空。设置 transferlen=15，TX 发送 16 个数据，RX 会接收到 15 个数据。多出的一个数据在 TX FIFO 中，获取 TX FIFO COUNT 的值为默认值-1；Disable 之后重新使能 TX/RX，TX FIFO 中剩下的数据会被发出。

### 16.3.15 TX/RX FIFO 请求中断

当 `uart_fifo_config_1` 中 `tx_fifo_cnt` 大于 `tx_fifo_th` 时, 产生 TX FIFO 请求中断。当条件不满足时该中断标志会自动清除。当 `uart_fifo_config_1` 中 `rx_fifo_cnt` 大于 `rx_fifo_th` 时, 产生 RX FIFO 请求中断。当条件不满足时该中断标志会自动清除。TX/RX 均可支持多次发送/接收, 并非一次性达到 `tx_fifo_th/rx_fifo_th` 设置的值。

例如设置寄存器 `uart_fifo_config_1` 中 `tx_fifo_th/rx_fifo_th`, 将 `rx_fifo_th/rx_fifo_th` 设置为 16; 设置寄存器 `utx_config` 中 `cr_utx_frm_en`, 使能 free run mode; 设置寄存器 `uart_int_mask` 中 `cr_utx_frdy_mask/cr_urx_frdy_mask` 为 0, 将 TX/RX 的 FIFO 中断打开; 设置寄存器 `utx_config/urx_config` 中 `cr_utx_en/cr_urx_en`, 使能 TX/RX; TX FIFO 中断: TX 会一直进入 FIFO 中断, 当芯片发送 128 字节后, 将 `cr_utx_frdy_mask` 置 1 屏蔽该中断。如果想再次进入 TX FIFO 中断, `cr_utx_frdy_mask` 置 0 即可。RX FIFO 中断: 对方先发送 15 字节数, 无中断产生, 此时获取 `rx_fifo_cnt` 值为 15, 再次发送 1 个字节达到 `rx_fifo_th` 设置的值时产生中断。产生发送中断后, 对方再次发送数据, 芯片可以收到数据。

### 16.3.16 RX 超时中断

RX 超时中断会在接收器超过超时门限值未收到数据时触发。门限值的超时单位是以通信的 Bit 为单位。

当对方给芯片发送数据时, 达到设置的超时单位后, 会产生一次超时中断。例如通过设置寄存器 `uart_int_mask` 中的 `cr_urx_rto_mask` 位为 0 将超时中断打开, 在超时中断中接收数据, 通过打印 RX 的 `buffer` 及打印的 `log`, 确认进入超时中断并且接受的数据正确。

### 16.3.17 RX 奇偶校验错误中断

RX 奇偶校验错误中断会发生在奇偶校验出错时。但是不影响 RX 正确接收并解析对方发来的数据。RX 在接收数据的时候, 会取前 8bit 作为数据位, 忽略奇偶校验位, 因此数据一致性完整。

例如设置寄存器 `utx_config/urx_config` 中 `cr_utx_prt_en/cr_urx_prt_en` 使能奇偶校验; 设置寄存器 `utx_config/urx_config` 中 `cr_utx_prt_sel/cr_urx_prt_sel` 选择奇偶校验种类; 当对方使用奇校验/偶校验给芯片发送数据时, 芯片的 RX 不开启奇偶校验, RX 收到的数据正确。

### 16.3.18 TX/RX FIFO 溢出中断

如果 TX/RX FIFO 发生了上溢或者下溢, 会触发对应的溢出中断, 当 FIFO 清除位: 寄存器 `UART_FIFO_CONFIG_0` 中 `tx_fifo_clr` 和 `rx_fifo_clr` 被置 1 时, 对应的 FIFO 会被清空, 同时溢出中断标志会自动清除。可以通过寄存器 `UART_INT_STS` 查询各中断状态, 通过向寄存器 `UART_INT_CLEAR` 相应的位写 1 清除中断。

### 16.3.19 RX BCR 中断

当 RX 接收到的数据达到寄存器 `urx_bcr_int_cfg` 中 `cr_urx_bcr_value` 设置的值时, 产生 BCR 中断。

与 RX END 中断的区别在于: END 中断适合接收已知长度的数据, BCR 中断则可用来接收未知长度的中断。END 中断的触发位置由 `cr_urx_len` 控制, 中断触发时 `counter` 就会清 0。BCR 中断的触发位置由 `cr_urx_bcr_value` 控制, 中断触发时 `counter` 不会清 0 而会继续累加, 但可由软件清 0 (`cr_urx_bcr_clr`) BCR 中断使用的场景配合链式 DMA 使用时, 不清楚当前 DMA 已经传输的数据量, 此时可通过 `count` 获知当前已知传输的数据量。

### 16.3.20 LIN 同步错误中断

当 utx\_config 中 cr\_utx\_lin\_en 使能后, 进入 LIN 模式, 在 LIN 模式下, 如果接收数据时, 没有检测到 LIN 总线的同步段, 则产生 LIN 同步错误中断。

### 16.3.21 通用/固定字符模式自动波特率检测中断

在自动波特率模式下, 根据用户的配置, 在检测到波特率时, 可以产生通用模式自动波特率检测中断或者固定字符模式自动波特率检测中断。

## 16.4 寄存器描述

名称	描述
utx_config	
urx_config	
uart_bit_prd	
data_config	
utx_ir_position	
urx_ir_position	
urx_rto_timer	
uart_sw_mode	
uart_int_sts	UART interrupt status
uart_int_mask	UART interrupt mask
uart_int_clear	UART interrupt clear
uart_int_en	UART interrupt enable
uart_status	
sts_urx_abr_prd	
urx_abr_prd_b01	
urx_abr_prd_b23	
urx_abr_prd_b45	
urx_abr_prd_b67	
urx_abr_pw_tol	
urx_bcr_int_cfg	
utx_rs485_cfg	
uart_fifo_config_0	
uart_fifo_config_1	

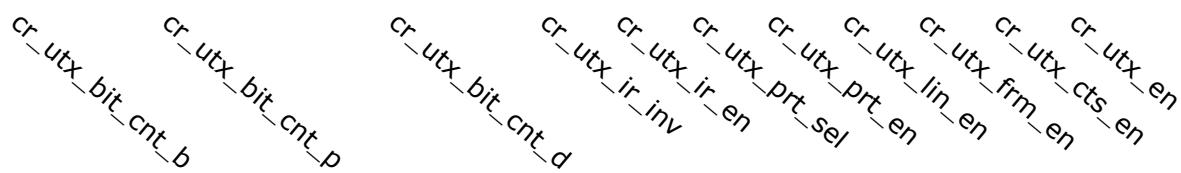
名称	描述
uart_fifo_wdata	
uart_fifo_rdata	

### 16.4.1 utx\_config

地址: 0x30002000

cr\_utx\_len

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

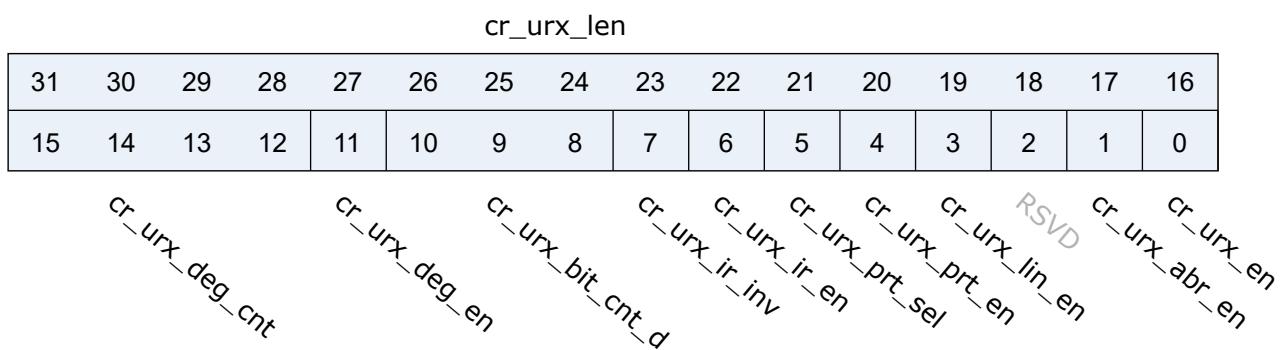


位	名称	权限	复位值	描述
31:16	cr_utx_len	r/w	16'd0	Length of UART TX data transfer (Unit: character/byte) (Don't-care if cr_utx_frm_en is enabled)
15:13	cr_utx_bit_cnt_b	r/w	3'd4	UART TX BREAK bit count (for LIN protocol) Note: Additional 8 bit times will be added since LIN Break field requires at least 13 bit times
12:11	cr_utx_bit_cnt_p	r/w	2'd1	UART TX STOP bit count (unit: 0.5 bit)
10:8	cr_utx_bit_cnt_d	r/w	3'd7	UART TX DATA bit count for each character
7	cr_utx_ir_inv	r/w	1'b0	Inverse signal of UART TX output in IR mode
6	cr_utx_ir_en	r/w	1'b0	Enable signal of UART TX IR mode
5	cr_utx_prt_sel	r/w	1'b0	Select signal of UART TX parity bit 1: Odd parity 0: Even parity
4	cr_utx_prt_en	r/w	1'b0	Enable signal of UART TX parity bit
3	cr_utx_lin_en	r/w	1'b0	Enable signal of UART TX LIN mode (LIN header will be sent before sending data)

位	名称	权限	复位值	描述
2	cr_utx_frm_en	r/w	1'b0	Enable signal of UART TX freerun mode (utx_end_int will be disabled)
1	cr_utx_cts_en	r/w	1'b0	Enable signal of UART TX CTS flow control function
0	cr_utx_en	r/w	1'b0	Enable signal of UART TX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

#### 16.4.2 urx\_config

地址: 0x30002004



位	名称	权限	复位值	描述
31:16	cr_urx_len	r/w	16'd0	Length of UART RX data transfer (Unit: character/byte) urx_end_int will assert when this length is reached
15:12	cr_urx_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_urx_deg_en	r/w	1'b0	Enable signal of RXD input de-glitch function
10:8	cr_urx_bit_cnt_d	r/w	3'd7	UART RX DATA bit count for each character
7	cr_urx_ir_inv	r/w	1'b0	Inverse signal of UART RX input in IR mode
6	cr_urx_ir_en	r/w	1'b0	Enable signal of UART RX IR mode
5	cr_urx_prt_sel	r/w	1'b0	Select signal of UART RX parity bit 1: Odd parity 0: Even parity
4	cr_urx_prt_en	r/w	1'b0	Enable signal of UART RX parity bit

位	名称	权限	复位值	描述
3	cr_urx_lin_en	r/w	1'b0	Enable signal of UART RX LIN mode (LIN header will be required and checked before receiving data)
2	RSVD			
1	cr_urx_abr_en	r/w	1'b0	Enable signal of UART RX Auto Baud Rate detection function
0	cr_urx_en	r/w	1'b0	Enable signal of UART RX function

### 16.4.3 uart\_bit\_prd

地址: 0x30002008

## cr\_urx\_bit\_prd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### cr\_utx\_bit\_prd

位	名称	权限	复位值	描述
31:16	cr_urx_bit_prd	r/w	16'd255	Period of each UART RX bit, related to baud rate
15:0	cr_utx_bit_prd	r/w	16'd255	Period of each UART TX bit, related to baud rate

#### 16.4.4 data\_config

地址: 0x3000200c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

位	名称	权限	复位值	描述
31:1	RSVD			
0	cr_uart_bit_inv	r/w	1'b0	Bit-inverse signal for each data byte 0: Each byte is sent out LSB-first 1: Each byte is sent out MSB-first

## 16.4.5 utx\_ir\_position

地址: 0x30002010

cr_utx_ir_pos_p															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_utx_ir_pos_s															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:16	cr_utx_ir_pos_p	r/w	16'd159	STOP position of UART TX IR pulse
15:0	cr_utx_ir_pos_s	r/w	16'd112	START position of UART TX IR pulse

## 16.4.6 urx\_ir\_position

地址: 0x30002014

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

cr\_urx\_ir\_pos\_s

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	cr_urx_ir_pos_s	r/w	16'd111	START position of UART RXD pulse recovered from IR signal

### 16.4.7 urx\_rto\_timer

地址: 0x30002018

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

cr\_urx\_rto\_value

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	cr_urx_rto_value	r/w	8'd15	Time-out value for triggering RTO interrupt (unit: bit time)

### 16.4.8 uart\_sw\_mode

地址: 0x3000201c

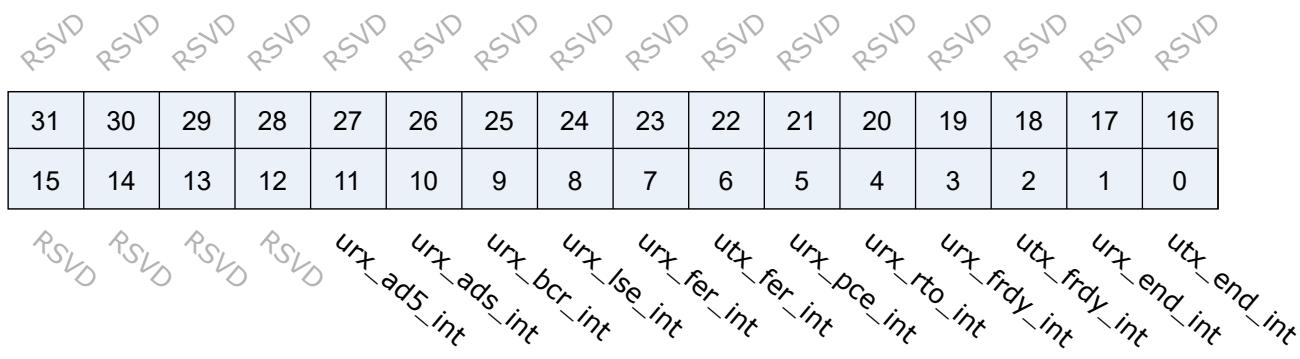
RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

cr\_urx\_rts\_sw\_val  
cr\_urx\_rts\_sw\_mode  
cr\_utx\_txd\_sw\_val  
cr\_utx\_txd\_sw\_mode

位	名称	权限	复位值	描述
31:4	RSVD			
3	cr_urx_rts_sw_val	r/w	1'b0	UART RX RTS output SW control value
2	cr_urx_rts_sw_mode	r/w	1'b0	UART RX RTS output SW control mode
1	cr_utx_txd_sw_val	r/w	1'b0	UART TX TXD output SW control value
0	cr_utx_txd_sw_mode	r/w	1'b0	UART TX TXD output SW control mode

### 16.4.9 uart\_int\_sts

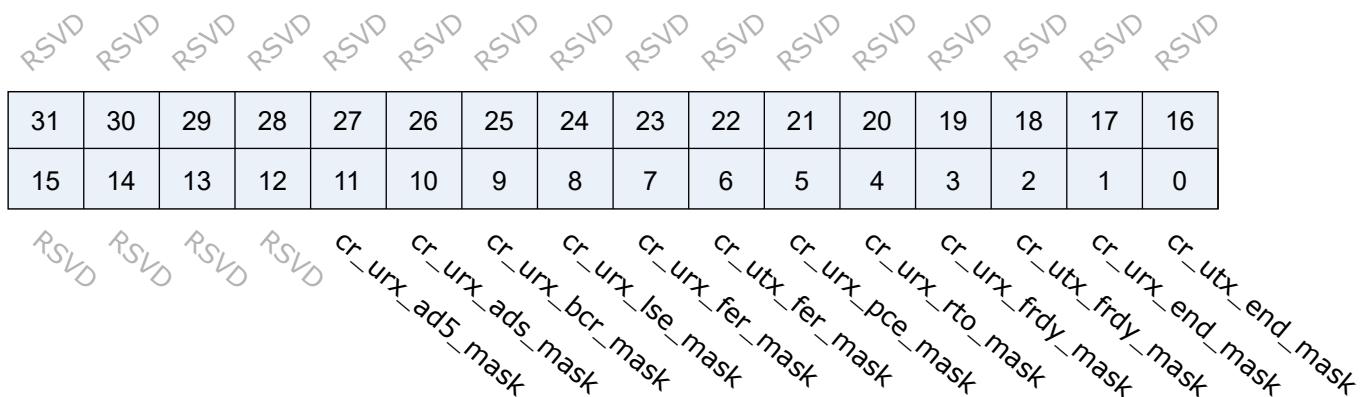
地址: 0x30002020



位	名称	权限	复位值	描述
31:12	RSVD			
11	urx_ad5_int	r	1'b0	UART RX ABR Detection finish interrupt using codeword 0x55
10	urx_ads_int	r	1'b0	UART RX ABR Detection finish interrupt using START bit
9	urx_bcr_int	r	1'b0	UART RX byte count reached interrupt
8	urx_lse_int	r	1'b0	UART RX LIN mode sync field error interrupt
7	urx_fer_int	r	1'b0	UART RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
6	utx_fer_int	r	1'b0	UART TX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
5	urx_pce_int	r	1'b0	UART RX parity check error interrupt
4	urx_rto_int	r	1'b0	UART RX Time-out interrupt
3	urx_frdy_int	r	1'b0	UART RX FIFO ready ( $rx\_fifo\_cnt > rx\_fifo\_th$ ) interrupt, auto-cleared when data is popped
2	utx_frdy_int	r	1'b1	UART TX FIFO ready ( $tx\_fifo\_cnt > tx\_fifo\_th$ ) interrupt, auto-cleared when data is pushed
1	urx_end_int	r	1'b0	UART RX transfer end interrupt (set according to cr_urx_len)
0	utx_end_int	r	1'b0	UART TX transfer end interrupt (set according to cr_utx_len)

### 16.4.10 uart\_int\_mask

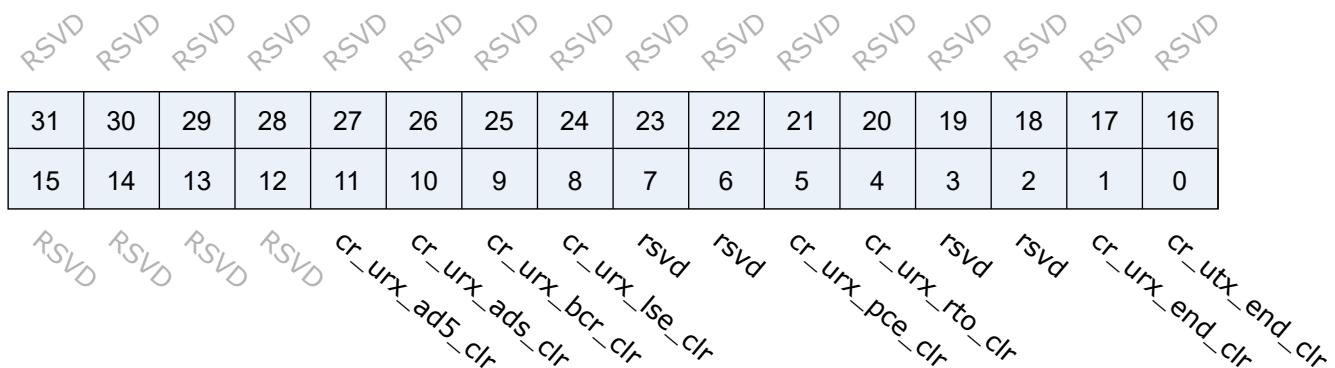
地址: 0x30002024



位	名称	权限	复位值	描述
31:12	RSVD			
11	cr_urx_ad5_mask	r/w	1'b1	Interrupt mask of urx_ad5_int
10	cr_urx_ads_mask	r/w	1'b1	Interrupt mask of urx_ads_int
9	cr_urx_bcr_mask	r/w	1'b1	Interrupt mask of urx_bcr_int
8	cr_urx_lse_mask	r/w	1'b1	Interrupt mask of urx_lse_int
7	cr_urx_fer_mask	r/w	1'b1	Interrupt mask of urx_fer_int
6	cr_utx_fer_mask	r/w	1'b1	Interrupt mask of utx_fer_int
5	cr_urx_pce_mask	r/w	1'b1	Interrupt mask of urx_pce_int
4	cr_urx_rto_mask	r/w	1'b1	Interrupt mask of urx_rto_int
3	cr_urx_frdy_mask	r/w	1'b1	Interrupt mask of urx_frdy_int
2	cr_utx_frdy_mask	r/w	1'b1	Interrupt mask of utx_frdy_int
1	cr_urx_end_mask	r/w	1'b1	Interrupt mask of urx_end_int
0	cr_utx_end_mask	r/w	1'b1	Interrupt mask of utx_end_int

### 16.4.11 uart\_int\_clear

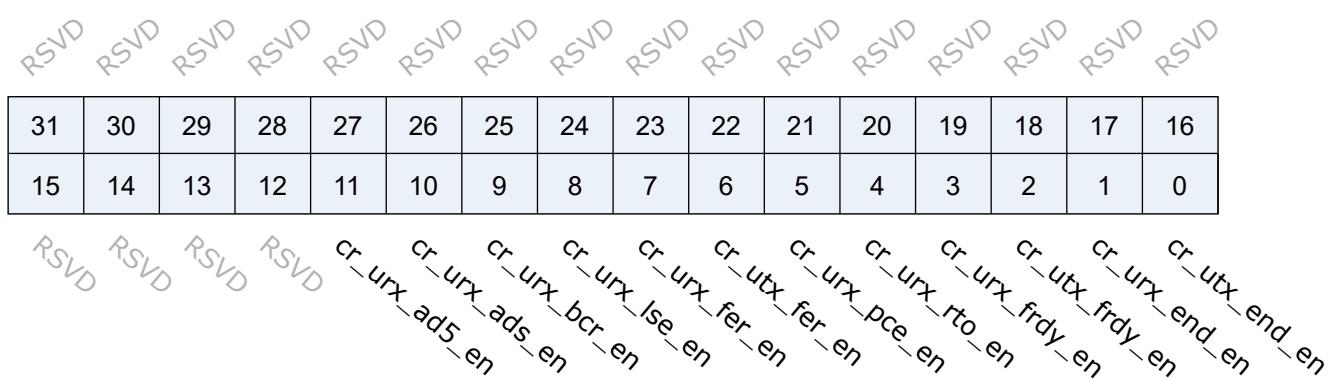
地址: 0x30002028



位	名称	权限	复位值	描述
31:12	RSVD			
11	cr_urx_ad5_clr	w1c	1'b0	Interrupt clear of urx_ad5_int
10	cr_urx_ads_clr	w1c	1'b0	Interrupt clear of urx_ads_int
9	cr_urx_bcr_clr	w1c	1'b0	Interrupt clear of urx_bcr_int
8	cr_urx_lse_clr	w1c	1'b0	Interrupt clear of urx_lse_int
7	rsvd	rsvd	1'b0	
6	rsvd	rsvd	1'b0	
5	cr_urx_pce_clr	w1c	1'b0	Interrupt clear of urx_pce_int
4	cr_urx_rto_clr	w1c	1'b0	Interrupt clear of urx_rto_int
3	rsvd	rsvd	1'b0	
2	rsvd	rsvd	1'b0	
1	cr_urx_end_clr	w1c	1'b0	Interrupt clear of urx_end_int
0	cr_utx_end_clr	w1c	1'b0	Interrupt clear of utx_end_int

### 16.4.12 uart\_int\_en

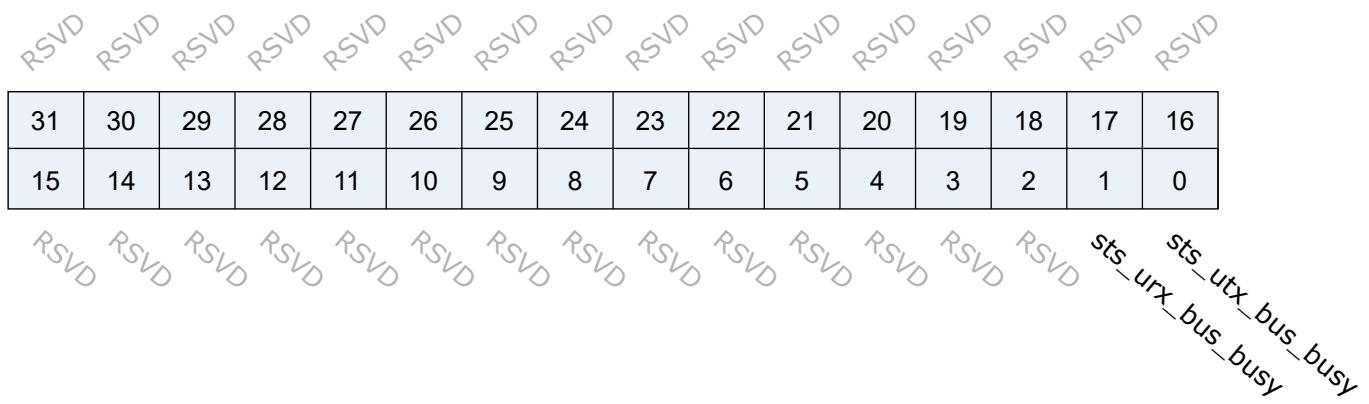
地址: 0x3000202c



位	名称	权限	复位值	描述
31:12	RSVD			
11	cr_urx_ad5_en	r/w	1'b1	Interrupt enable of urx_ad5_int
10	cr_urx_ads_en	r/w	1'b1	Interrupt enable of urx_ads_int
9	cr_urx_bcr_en	r/w	1'b1	Interrupt enable of urx_bcr_int
8	cr_urx_lse_en	r/w	1'b1	Interrupt enable of urx_lse_int
7	cr_urx_fer_en	r/w	1'b1	Interrupt enable of urx_fer_int
6	cr_utx_fer_en	r/w	1'b1	Interrupt enable of utx_fer_int
5	cr_urx_pce_en	r/w	1'b1	Interrupt enable of urx_pce_int
4	cr_urx_rto_en	r/w	1'b1	Interrupt enable of urx_rto_int
3	cr_urx_frdy_en	r/w	1'b1	Interrupt enable of urx_frdy_int
2	cr_utx_frdy_en	r/w	1'b1	Interrupt enable of utx_frdy_int
1	cr_urx_end_en	r/w	1'b1	Interrupt enable of urx_end_int
0	cr_utx_end_en	r/w	1'b1	Interrupt enable of utx_end_int

### 16.4.13 uart\_status

地址: 0x30002030



位	名称	权限	复位值	描述
31:2	RSVD			
1	sts_urx_bus_busy	r	1'b0	Indicator of UART RX bus busy
0	sts_utx_bus_busy	r	1'b0	Indicator of UART TX bus busy

### 16.4.14 sts\_urx\_abr\_prd

地址: 0x30002034

sts\_urx\_abr\_prd\_0x55

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_urx\_abr\_prd\_start

位	名称	权限	复位值	描述
31:16	sts_urx_abr_prd_0x55	r	16'd0	Bit period of Auto Baud Rate detection using codeword 0x55
15:0	sts_urx_abr_prd_start	r	16'd0	Bit period of Auto Baud Rate detection using START bit

### 16.4.15 urx\_abr\_prd\_b01

地址: 0x30002038

sts\_urx\_abr\_prd\_bit1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_urx\_abr\_prd\_bit0

位	名称	权限	复位值	描述
31:16	sts_urx_abr_prd_bit1	r	16'd0	Bit period of Auto Baud Rate detection - bit[1]
15:0	sts_urx_abr_prd_bit0	r	16'd0	Bit period of Auto Baud Rate detection - bit[0]

### 16.4.16 urx\_abr\_prd\_b23

地址: 0x3000203c

sts\_urx\_abr\_prd\_bit3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_urx\_abr\_prd\_bit2

位	名称	权限	复位值	描述
31:16	sts_urx_abr_prd_bit3	r	16'd0	Bit period of Auto Baud Rate detection - bit[3]
15:0	sts_urx_abr_prd_bit2	r	16'd0	Bit period of Auto Baud Rate detection - bit[2]

### 16.4.17 urx\_abr\_prd\_b45

地址: 0x30002040

sts\_urx\_abr\_prd\_bit5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_urx\_abr\_prd\_bit4

位	名称	权限	复位值	描述
31:16	sts_urx_abr_prd_bit5	r	16'd0	Bit period of Auto Baud Rate detection - bit[5]
15:0	sts_urx_abr_prd_bit4	r	16'd0	Bit period of Auto Baud Rate detection - bit[4]

### 16.4.18 urx\_abr\_prd\_b67

地址: 0x30002044

sts\_urx\_abr\_prd\_bit7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts\_urx\_abr\_prd\_bit6

位	名称	权限	复位值	描述
31:16	sts_urx_abr_prd_bit7	r	16'd0	Bit period of Auto Baud Rate detection - bit[7]
15:0	sts_urx_abr_prd_bit6	r	16'd0	Bit period of Auto Baud Rate detection - bit[6]

### 16.4.19 urx\_abr\_pw\_tol

地址: 0x30002048

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_urx\_abr\_pw\_tol

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	cr_urx_abr_pw_tol	r/w	8'd3	Auto Baud Rate detection pulse-width tolerance for using codeword 0x55

### 16.4.20 urx\_bcr\_int\_cfg

地址: 0x30002050

sts\_urx\_bcr\_count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_urx\_bcr\_value

位	名称	权限	复位值	描述
31:16	sts_urx_bcr_count	r	16'd0	Current byte count of urx_bcr_int counter, auto-cleared by cr_urx_bcr_clr
15:0	cr_urx_bcr_value	r/w	16'hFFFF	Byte count setting for urx_bcr_int counter

## 16.4.21 utx\_rs485\_cfg

地址: 0x30002054

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:2	RSVD			
1	cr_utx_rs485_pol	r/w	1'b1	DE pin polarity in RS-485 transceiver mode 1'b0: DE is active-low 1'b1: DE is active-high
0	cr_utx_rs485_en	r/w	1'b0	Enable signal for RS-485 transceiver mode 1'b0: Disabled, normal UART 1'b1: Enabled, IO is connected to RS-485 transceiver and RTS_O becomes DE function

## 16.4.22 uart\_fifo\_config\_0

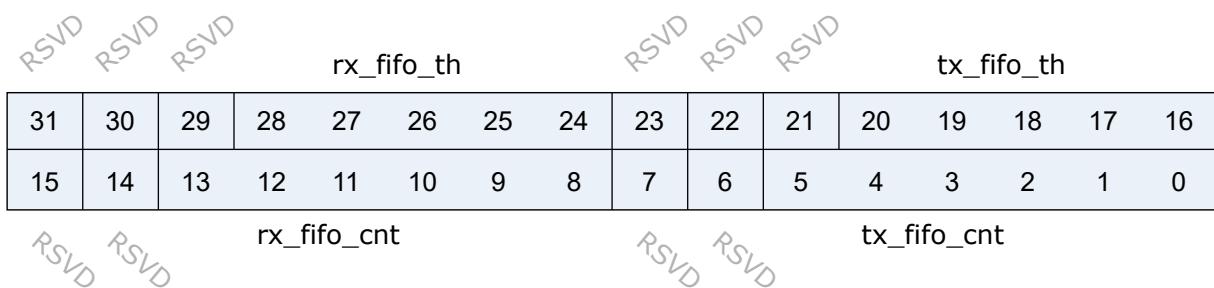
地址: 0x30002080

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

位	名称	权限	复位值	描述
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	uart_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	uart_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

#### 16.4.23 uart\_fifo\_config\_1

地址: 0x30002084



位	名称	权限	复位值	描述
31:29	RSVD			
28:24	rx_fifo_th	r/w	5'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:21	RSVD			
20:16	tx_fifo_th	r/w	5'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:14	RSVD			
13:8	rx_fifo_cnt	r	6'd0	RX FIFO available count
7:6	RSVD			

位	名称	权限	复位值	描述
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count

#### 16.4.24 uart\_fifo\_wdata

地址: 0x30002088

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
uart_fifo_wdata															

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	uart_fifo_wdata	w	x	

#### 16.4.25 uart\_fifo\_rdata

地址: 0x3000208c

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
uart_fifo_rdata															

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	uart_fifo_rdata	r	8'h0	

## 17.1 简介

I2C (Inter-Integrated Circuit) 是一种串行通讯总线，使用多主从架构，用来连接低速外围装置。每个器件都有一个唯一的地址识别，并且都可以作为一个发送器或接收器。每个连接到总线的器件都可以通过唯一的地址和一直存在的主、从机关系用软件设置地址，主机可以作为主机发送器或主机接收器。如果有两个或多个主机同时初始化，数据传输可以通过冲突检测和仲裁防止数据被破坏。BL808 包含 4 组 I2C 控制器主机，可灵活配置 slaveAddr、subAddr 以及传输数据，方便与从设备通信，提供 2 个 word 深度的 fifo，提供中断功能，可搭配 DMA 使用提高效率，可灵活调整时钟频率。

## 17.2 主要特征

- 支持主机模式
- 支持多主机模式和仲裁功能
- 时钟频率可灵活调整
- 支持 10 地址模式
- 支持 DMA 传输模式

## 17.3 功能描述

引脚列表：

表 17.1: I2C 引脚

名称	类型	描述
I2Cx_SCL	输入/输出	I2C 串行时钟信号
I2Cx_SDA	输入/输出	I2C 串行数据信号

### 17.3.1 起始和停止条件

所有传输都由起始条件 (START condition) 开始, 以停止条件 (STOP condition) 结束。起始条件和停止条件一般都由主机产生, 总线在起始条件后被认为处于总线忙的状态, 在停止条件后的某段时间内被认为处于空闲状态。

起始条件:SCL 为高电平时 SDA 产生一高至低的电平转换;

停止条件:SCL 为高电平时 SDA 产生一低至高的电平转换。

波形示意图如下:

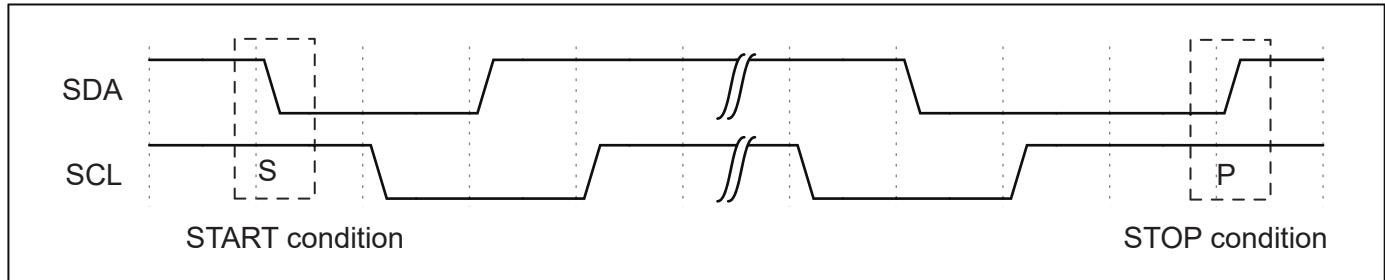


图 17.1: I2C 起始和停止条件

### 17.3.2 数据传输格式

7 地址模式:

传输的第一个 8 位为寻址字节, 包括 7 位从机地址和 1 位方向位。数据由主机发送或接收是由主机所送出的第 1 个字节的第 8 位控制, 若为 0 表示数据由主机发送; 为 1 则表示数据由主机接收, 紧接着从机发出应答位 (ACK), 在数据传输完成后, 主机发出停止信号, 波形图如下:

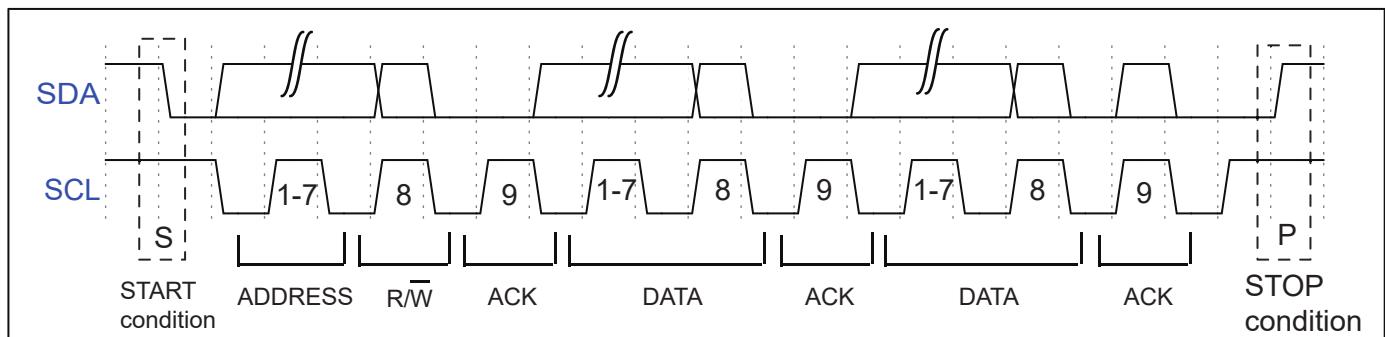


图 17.2: I2C 数据传输格式

主发送和从接收的时序

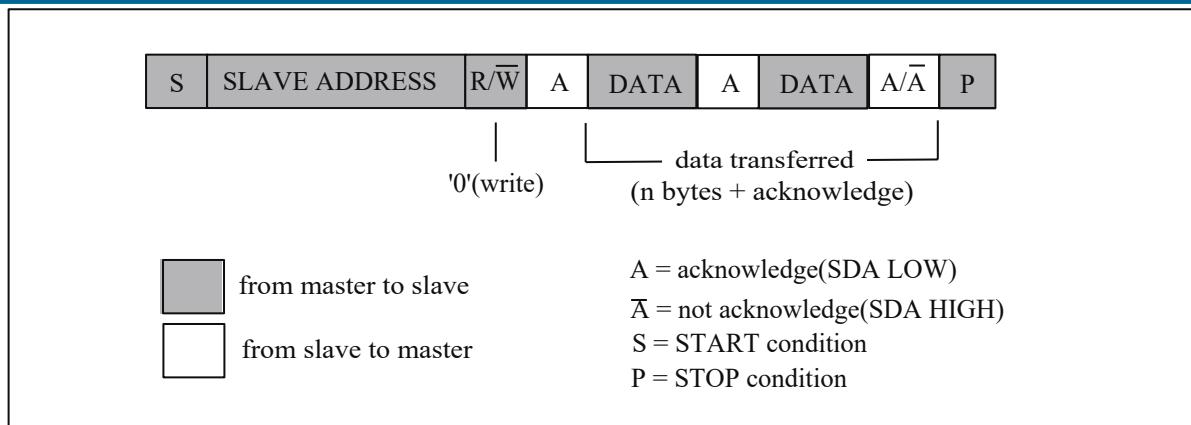


图 17.3: 主发送和从接收的时序

主接收和从发送的时序

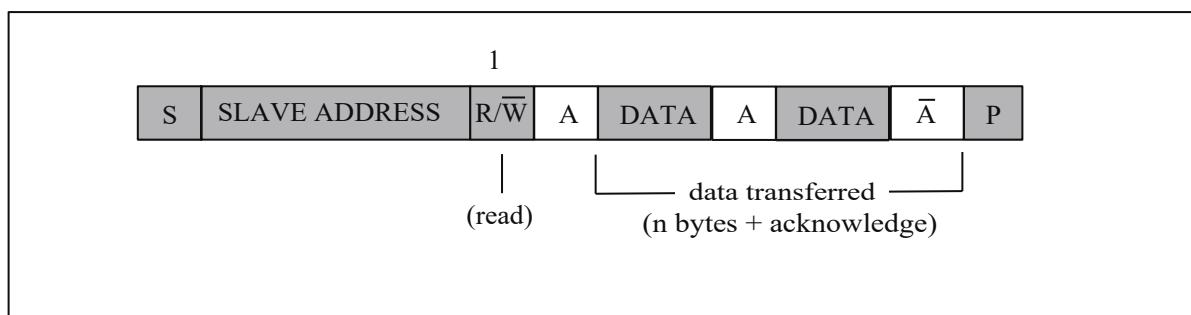


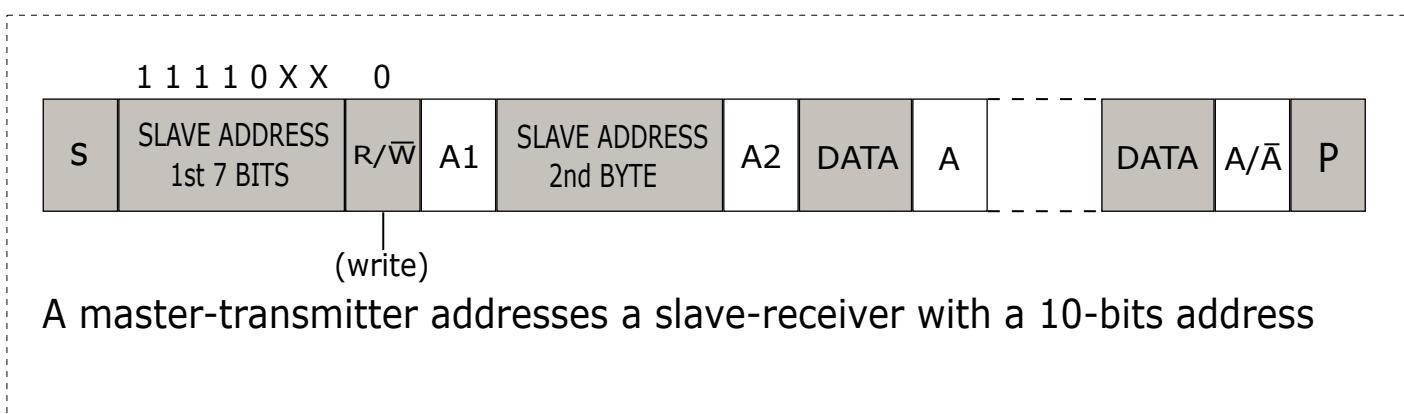
图 17.4: 主接收和从发送的时序

10 地址模式: 使用时需要将寄存器 i2c\_config 中 cr\_i2c\_10b\_addr\_en 置 1。

10bit 的从机地址由开始条件 (S) 或重复开始条件 (Sr) 后的两个字节组成。第一个字节的前 7 位是 1111 0XX，XX 是 10bit 地址的最高有效位的前两位。第一个字节的第 8bit 是读写位，决定传输方向。尽管 1111 XXX 有 8 种可能的组合，然后只有 1111 0XX 这四种可以用于 10bit 寻址，剩下的 1111 1XX 这四种是为将来 I2C 扩展用的。

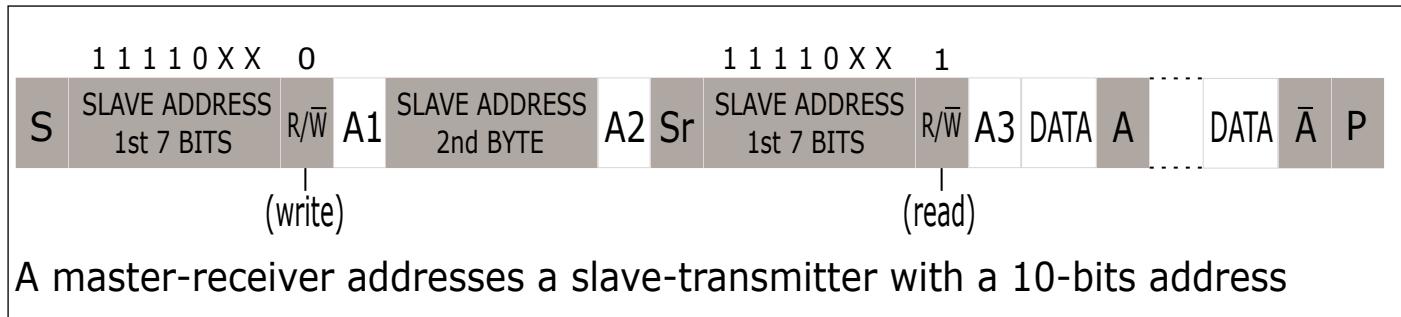
前面描述的用于 7bit 寻址的读写格式都适用于 10bit 寻址，详情如下:

#### 1. 主-发送器传输到从-接收器 (10bit 从机地址)



从图中看出传输方向不变。当接收到开始条件后的 10bit 地址，从机就和它自己的地址比较从机地址的第一个字节 (1111 0XX)，并检查第八个 bit(读写位)是否为 0。有可能多个设备都匹配并产生应答 (A1)。接下来所有从机开始匹配自己地址与第二个字节的 8 个 bit(XXXX XXXX)，这时就只有一个从机匹配并产生应答 (A2)，被主机寻址匹配的从机会保持被寻址的状态直到接收到终止条件或者是重复开始条件后跟着一个不同的从机地址。

## 2. 主-接收器从从-发送器接收数据 (10bit 从地址)



在第二个读写位之后传输方向就会改变。在第二个应答 A2 之前，处理过程与上面的主-发送器寻址从-接收器一致。在重复开始条件 (Sr) 之后，匹配的从机会保持被寻址上的状态。这个从机会检查 Sr 之后的第一个字节的前 7bit 是否正确，然后测试第 8bit 是否为 1(读)。如果这也匹配的话，从机就认定它被作为一个发送器被寻址到了并产生应答 A3。从-发送器会保持被寻址的状态直到接收到终止条件 (P) 或者重复开始条件 (Sr) 跟着一个不同的从机地址，然后这个时候的重复开始条件下，所有的从机会比较它们的地址与 11110XX 比较并测试第八位 (读写位)。然而它们不会寻址到，因为对于 10bit 设备，读写位是 1，或者对于 7bit 的设备，1111 0XX 的从机地址不匹配。

### 17.3.3 仲裁

当 I2C 总线存在多个主机时，可能会发生多个主机同时启动传输的情况，此时必须要依靠仲裁机制来决定哪个主机有权利完成接下来的数据传输，其余主机则须放弃对总线的控制，等到总线再次空出来后才能再次启动传输。

在传输过程中，所有主机都需要在 SCL 为高电平时检查 SDA 是否与自己所想送出的资料相符，当 SDA 电平与预期不同时，表示有别的主机也在同时进行传输，而发现 SDA 电平不同的主机则失去此次仲裁，由其他主机完成数据传输。

两主机同时传输数据并启动仲裁机制的波形示意图如下：

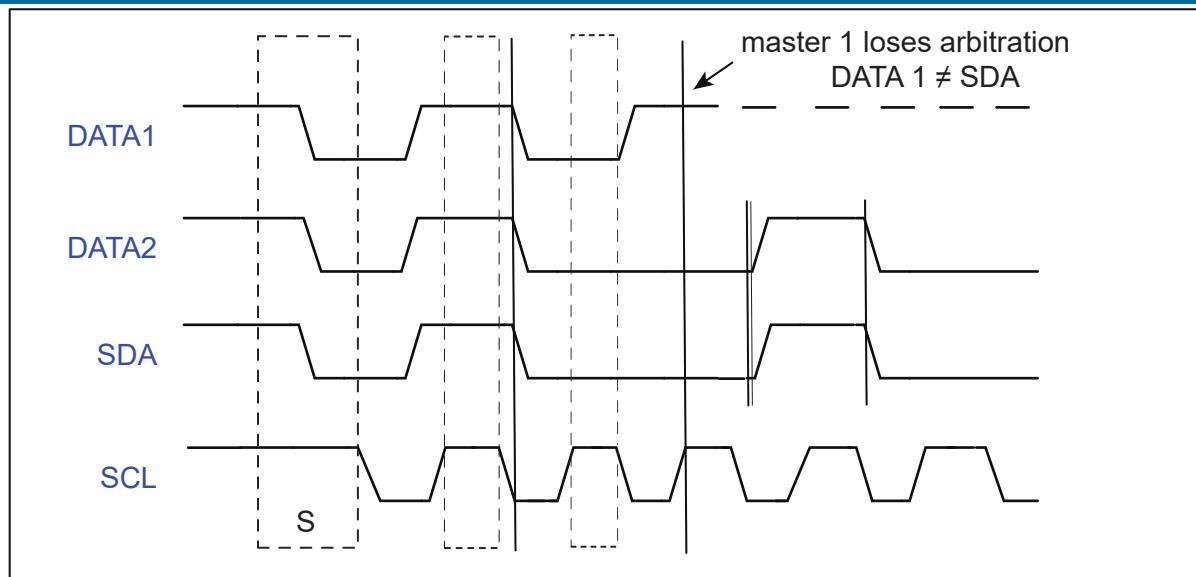


图 17.5: 同时传输数据波形示意图

## 17.4 I2C 时钟设定

I2C 的时钟可由 `bclk(bus clock)` 和 `xclk` 而来，可以在其的基础上做分频处理。寄存器 `i2c_prd_data` 可以对数据段的时钟做分频处理。i2c 模块将数据发送分为 4 个阶段，每个阶段在寄存器中用单独一个字节来控制，每个阶段的采样个数是可以设置的，4 个采样数共同决定了 `i2c_clock` 的分频系数。比如现在 `bclk` 是 32M，寄存器 `i2c_prd_data` 在不做配置默认情况下的值是 `0x0f0f0f0f`，那么 I2C 的时钟频率为  $32M / ((15 + 1) * 4) = 500K$ 。同理，寄存器 `i2c_prd_start` 和 `i2c_prd_stop` 也会分别对起始位和停止位的时钟做分频处理。

## 17.5 I2C 配置流程

### 17.5.1 配置项

- 读写标志位
- 从设备地址
- 从设备寄存器地址
- 从设备寄存器地址长度
- 数据 (发送时，配置发送的数据；接收时，存储接收到的数据)
- 数据长度
- 使能信号

## 17.5.2 读写标志位

I2C 支持发送和接收两种工作状态，寄存器 i2c\_config 中 cr\_i2c\_pkt\_dir 表示发送或者接收状态，设置为 0 时，表示发送状态，设置为 1 时，表示接收状态。

## 17.5.3 从设备地址

每个对接 I2C 的从设备，都会有唯一设备地址，通常该地址是 7 位长度，将从设备地址写入寄存器 i2c\_config 中 cr\_i2c\_slv\_addr，I2C 在将从设备地址发送出去之前，会自动左移 1 位，并在最低位补上发送接收方向位。

## 17.5.4 从设备寄存器地址

从设备寄存器地址表示 I2C 需要对从设备某个寄存器做读写操作的寄存器地址。将从设备寄存器地址写入寄存器 i2c\_sub\_addr，同时需要将寄存器 i2c\_config 中 cr\_i2c\_sub\_addr\_en 置 1。如果将寄存器 i2c\_config 中 cr\_i2c\_sub\_addr\_en 置 0，那么 I2C 主机发送时会跳过从设备寄存器地址段。

## 17.5.5 从设备寄存器地址长度

将从设备寄存器地址长度减 1 再写入寄存器 i2c\_config 中 cr\_i2c\_sub\_addr\_bc。

## 17.5.6 数据

数据部分表示需要发送到从设备的数据，或者需要从从设备接收到的数据。当 I2C 发送数据时，需要将数据依次以 word 为单位写入 I2C FIFO，发送数据写 FIFO 的寄存器地址 i2c\_fifo\_wdata。当 I2C 接收数据时，需要依次以 word 为单位从 I2C FIFO 中将数据读出来，接收数据读 FIFO 的寄存器地址 i2c\_fifo\_rdata。

## 17.5.7 数据长度

将数据长度减 1 再写入寄存器 i2c\_config 中 cr\_i2c\_pkt\_len。

## 17.5.8 使能信号

将以上几项配置完成后，再将使能信号寄存器 i2c\_config 中 cr\_i2c\_m\_en 置 1，就自动启动 I2C 发送流程了。

当读写标志位配置为 0 时，I2C 发送数据，主机发送流程：

1. 起始位
2. (从设备地址左移 1 位 + 0) + ACK
3. 从设备寄存器地址 + ACK
4. 1 字节数据 + ACK
5. 1 字节数据 + ACK
6. 停止位

当读写标志位配置为 1 时，I2C 接收数据，主机发送流程：

1. 起始位
2. (从设备地址左移 1 位 + 0) + ACK

3. 从设备寄存器地址 + ACK
4. 起始位
5. (从设备地址左移 1 位 + 1) + ACK
6. 1 字节数据 + ACK
7. 1 字节数据 + ACK
8. 停止位

## 17.6 FIFO 管理

I2C FIFO 深度为 2 个 word, I2C 发送和接收可分为 RX FIFO 和 TX FIFO。寄存器 i2c\_fifo\_config\_1 中 rx\_fifo\_cnt 表示 RX FIFO 中有多少数据 (单位 word) 需要读取。寄存器 i2c\_fifo\_config\_1 中 tx\_fifo\_cnt 表示 TX FIFO 中剩余多少空间 (单位 Word) 可供写入。

I2C FIFO 状态:

- RX FIFO underflow: 当 RX FIFO 中的数据被读取完毕或者为空时, 继续从 RX FIFO 中读取数据, 寄存器 i2c\_fifo\_config\_0 中 rx\_fifo\_underflow 会被置 1;
- RX FIFO overflow: 当 I2C 接收数据直到 RX FIFO 的 2 个 word 被填满后, 在没有读取 RX FIFO 的情况下, I2C 再次接收到数据, 寄存器 i2c\_fifo\_config\_0 中 rx\_fifo\_overflow 会被置 1;
- TX FIFO underflow: 当向 TX FIFO 中填入的数据大小不满足配置的 I2C 数据长度:i2c\_config 中 cr\_i2c\_pkt\_len, 并且已经没有新数据继续填入 TX FIFO 中时, 寄存器 i2c\_fifo\_config\_0 中 tx\_fifo\_underflow 会被置 1;
- TX FIFO overflow: 当 TX FIFO 的 2 个 word 被填满后, 在 TX FIFO 中的数据没有发出去之前, 再次向 TX FIFO 中填入数据, 寄存器 i2c\_fifo\_config\_0 中 tx\_fifo\_overflow 会被置 1。

## 17.7 搭配使用 DMA

I2C 可以使用 DMA 进行数据的发送和接收。将寄存器 i2c\_fifo\_config\_0 中 i2c\_dma\_tx\_en 置 1, 则开启 DMA 发送模式, 为 I2C 分配好通道后, DMA 会将数据从存储区传输到 i2c\_fifo\_wdata 寄存器中。将寄存器 i2c\_fifo\_config\_0 中 i2c\_dma\_rx\_en 置 1, 则开启 DMA 接收模式, 为 I2C 分配好通道后, DMA 会将 i2c\_fifo\_rdata 寄存器中的数据传输到存储区中。I2C 模块搭配使用 DMA 时, 数据部分将由 DMA 自动完成搬运, 不需要 CPU 再将数据写入 I2C TX FIFO 或者从 I2C RX FIFO 中读取数据。

### 17.7.1 DMA 发送流程

1. 配置读写标志位为 0
2. 配置从设备地址
3. 配置从设备寄存器地址
4. 配置从设备寄存器地址长度
5. 数据长度
6. 使能信号寄存器置 1
7. 配置 DMA transfer size
8. 配置 DMA 源地址 transfer width
9. 配置 DMA 目的地地址 transfer width(需要注意 I2C 搭配 DMA 使用时, 目的地地址 transfer width 需要设置为 32bits, 以 word 对齐使用)

10. 配置 DMA 源地址为存储发送数据的内存地址
11. 配置 DMA 目的地为 I2C TX FIFO 地址, i2c\_fifo\_wdata
12. 使能 DMA

## 17.7.2 DMA 接收流程

1. 配置读写标志位为 1
2. 配置从设备地址
3. 配置从设备寄存器地址
4. 配置从设备寄存器地址长度
5. 数据长度
6. 使能信号寄存器置 1
7. 配置 DMA transfer size
8. 配置 DMA 源地址 transfer width(需要注意 I2C 搭配 DMA 使用时, 源地址 transfer width 需要设置为 32bits, 以 word 对齐使用)
9. 配置 DMA 目的地 transfer width
10. 配置 DMA 源地址为 I2C RX FIFO 地址, i2c\_fifo\_rdata
11. 配置 DMA 目的地为存储接收数据的内存地址
12. 使能 DMA

## 17.8 中断

I2C 包括如下几种中断:

- I2C\_TRANS\_END\_INT: I2C 传输结束中断
- I2C\_TX\_FIFO\_READY\_INT: 当 I2C TX FIFO 有空闲空间可用于填充时, 触发中断
- I2C\_RX\_FIFO\_READY\_INT: 当 I2C RX FIFO 接收到数据时, 触发中断
- I2C\_NACK\_RECV\_INT: 当 I2C 模块检测到 NACK 状态, 触发中断
- I2C\_ARB\_LOST\_INT: I2C 仲裁丢失中断
- I2C\_FIFO\_ERR\_INT: I2C FIFO ERROR 中断

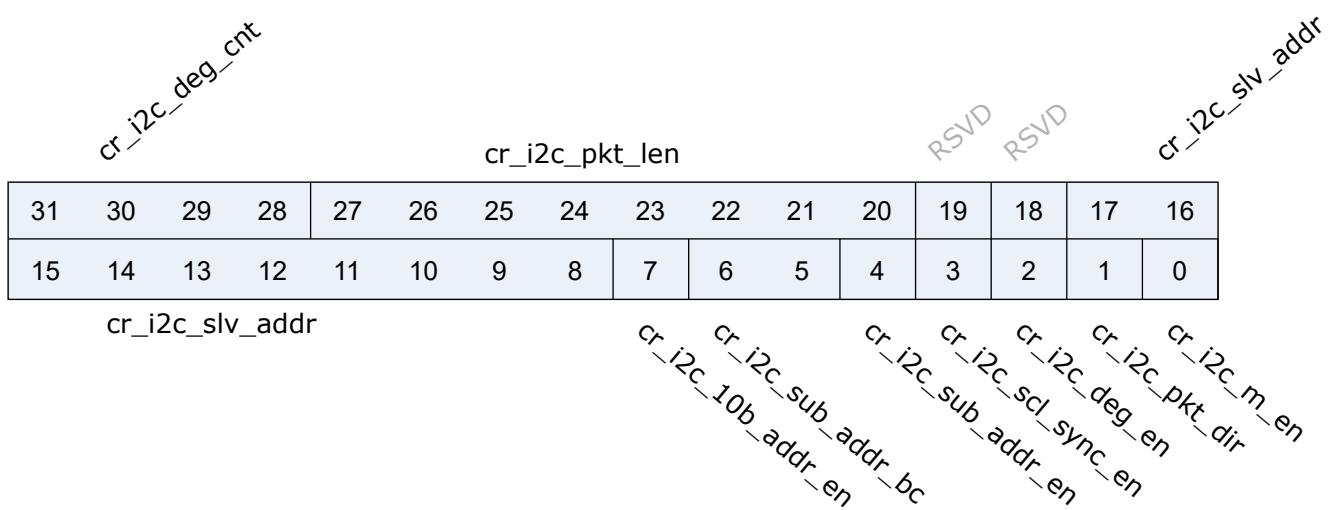
## 17.9 寄存器描述

名称	描述
i2c_config	
i2c_int_sts	
i2c_sub_addr	
i2c_bus_busy	

名称	描述
i2c_prd_start	
i2c_prd_stop	
i2c_prd_data	
i2c_fifo_config_0	
i2c_fifo_config_1	
i2c_fifo_wdata	
i2c_fifo_rdata	

### 17.9.1 i2c\_config

地址: 0x30003000

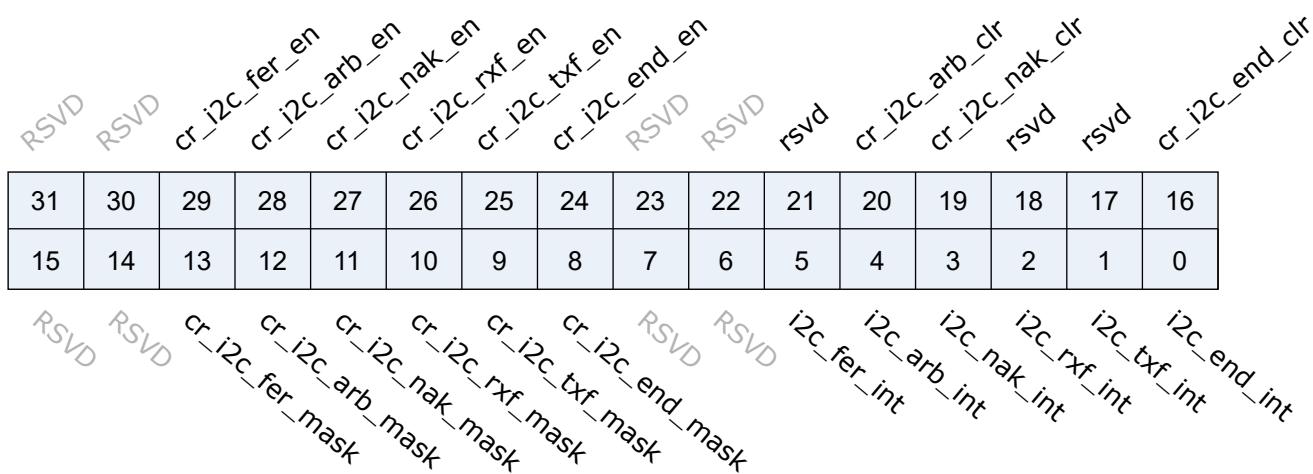


位	名称	权限	复位值	描述
31:28	cr_i2c_deg_cnt	r/w	4'd0	De-glitch function cycle count
27:20	cr_i2c_pkt_len	r/w	8'd0	Packet length (unit: byte)
19:18	RSVD			
17:8	cr_i2c_slv_addr	r/w	10'h0	Slave address for I2C transaction (target address)
7	cr_i2c_10b_addr_en	r/w	1'b0	Slave address 10-bit mode enable
6:5	cr_i2c_sub_addr_bc	r/w	2'd0	Sub-address field byte count 2'd0: 1-byte, 2'd1: 2-byte, 2'd2: 3-byte, 2'd3: 4-byte

位	名称	权限	复位值	描述
4	cr_i2c_sub_addr_en	r/w	1'b0	Enable signal of I2C sub-address field
3	cr_i2c_scl_sync_en	r/w	1'b1	Enable signal of I2C SCL synchronization, should be enabled to support Multi-Master and Clock-Stretching (Normally should not be turned-off)
2	cr_i2c_deg_en	r/w	1'b0	Enable signal of I2C input de-glitch function (for all input pins)
1	cr_i2c_pkt_dir	r/w	1'b1	Transfer direction of the packet 1'b0: Write; 1'b1: Read
0	cr_i2c_m_en	r/w	1'b0	Enable signal of I2C Master function Asserting this bit will trigger the transaction, and should be de-asserted after finish

### 17.9.2 i2c\_int\_sts

地址: 0x30003004



位	名称	权限	复位值	描述
31:30	RSVD			
29	cr_i2c_fer_en	r/w	1'b1	Interrupt enable of i2c_fer_int
28	cr_i2c_arb_en	r/w	1'b1	Interrupt enable of i2c_arb_int
27	cr_i2c_nak_en	r/w	1'b1	Interrupt enable of i2c_nak_int
26	cr_i2c_rxf_en	r/w	1'b1	Interrupt enable of i2c_rxf_int
25	cr_i2c_txf_en	r/w	1'b1	Interrupt enable of i2c_txf_int
24	cr_i2c_end_en	r/w	1'b1	Interrupt enable of i2c_end_int

位	名称	权限	复位值	描述
23:22	RSVD			
21	rsvd	rsvd	1'b0	
20	cr_i2c_arb_clr	w1c	1'b0	Interrupt clear of i2c_arb_int
19	cr_i2c_nak_clr	w1c	1'b0	Interrupt clear of i2c_nak_int
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_i2c_end_clr	w1c	1'b0	Interrupt clear of i2c_end_int
15:14	RSVD			
13	cr_i2c_fer_mask	r/w	1'b1	Interrupt mask of i2c_fer_int
12	cr_i2c_arb_mask	r/w	1'b1	Interrupt mask of i2c_arb_int
11	cr_i2c_nak_mask	r/w	1'b1	Interrupt mask of i2c_nak_int
10	cr_i2c_rxf_mask	r/w	1'b1	Interrupt mask of i2c_rxf_int
9	cr_i2c_txf_mask	r/w	1'b1	Interrupt mask of i2c_txf_int
8	cr_i2c_end_mask	r/w	1'b1	Interrupt mask of i2c_end_int
7:6	RSVD			
5	i2c_fer_int	r	1'b0	I2C TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
4	i2c_arb_int	r	1'b0	I2C arbitration lost interrupt
3	i2c_nak_int	r	1'b0	I2C NACK-received interrupt
2	i2c_rxf_int	r	1'b0	I2C RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
1	i2c_txf_int	r	1'b1	I2C TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	i2c_end_int	r	1'b0	I2C transfer end interrupt

### 17.9.3 i2c\_sub\_addr

地址: 0x30003008

cr\_i2c\_sub\_addr\_b3

cr\_i2c\_sub\_addr\_b2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_i2c\_sub\_addr\_b1

cr\_i2c\_sub\_addr\_b0

位	名称	权限	复位值	描述
31:24	cr_i2c_sub_addr_b3	r/w	8'd0	I2C sub-address field - byte[3]
23:16	cr_i2c_sub_addr_b2	r/w	8'd0	I2C sub-address field - byte[2]
15:8	cr_i2c_sub_addr_b1	r/w	8'd0	I2C sub-address field - byte[1]
7:0	cr_i2c_sub_addr_b0	r/w	8'd0	I2C sub-address field - byte[0] (sub-address starts from this byte)

#### 17.9.4 i2c\_bus\_busy

地址: 0x3000300c

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD   RSVD

cr\_i2c\_bus\_busy\_clr   sts\_i2c\_bus\_busy

位	名称	权限	复位值	描述
31:2	RSVD			
1	cr_i2c_bus_busy_clr	w1c	1'b0	Clear signal of bus_busy status, not for normal usage (in case I2C bus hangs)
0	sts_i2c_bus_busy	r	1'b0	Indicator of I2C bus busy

#### 17.9.5 i2c\_prd\_start

地址: 0x30003010

cr_i2c_prd_s_ph_3								cr_i2c_prd_s_ph_2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr\_i2c\_prd\_s\_ph\_1      cr\_i2c\_prd\_s\_ph\_0

位	名称	权限	复位值	描述
31:24	cr_i2c_prd_s_ph_3	r/w	8'd15	Length of START condition phase 3
23:16	cr_i2c_prd_s_ph_2	r/w	8'd15	Length of START condition phase 2
15:8	cr_i2c_prd_s_ph_1	r/w	8'd15	Length of START condition phase 1
7:0	cr_i2c_prd_s_ph_0	r/w	8'd15	Length of START condition phase 0

### 17.9.6 i2c\_prd\_stop

地址: 0x30003014

cr_i2c_prd_p_ph_3								cr_i2c_prd_p_ph_2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_i2c_prd_p_ph_1								cr_i2c_prd_p_ph_0							

位	名称	权限	复位值	描述
31:24	cr_i2c_prd_p_ph_3	r/w	8'd15	Length of STOP condition phase 3
23:16	cr_i2c_prd_p_ph_2	r/w	8'd15	Length of STOP condition phase 2
15:8	cr_i2c_prd_p_ph_1	r/w	8'd15	Length of STOP condition phase 1
7:0	cr_i2c_prd_p_ph_0	r/w	8'd15	Length of STOP condition phase 0

### 17.9.7 i2c\_prd\_data

地址: 0x30003018

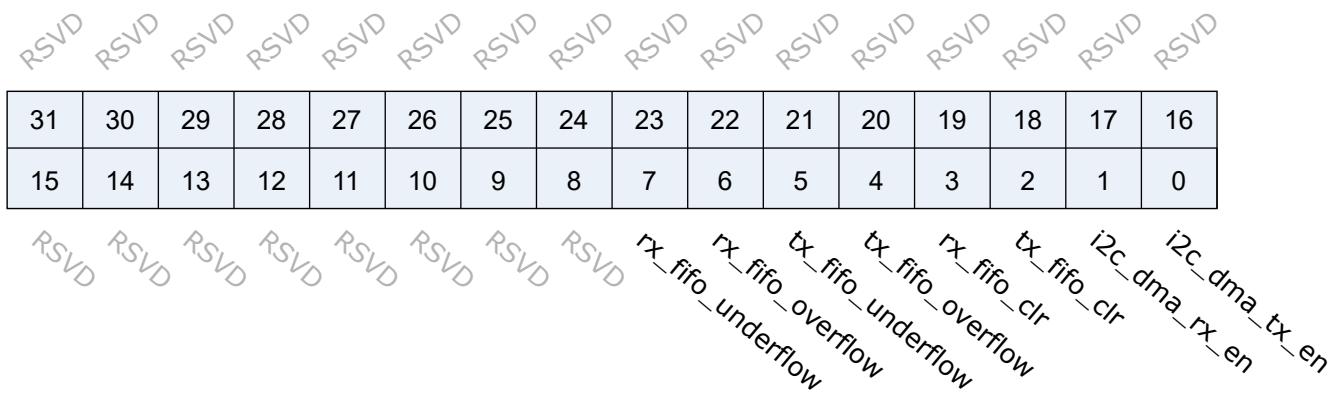
cr_i2c_prd_d_ph_3								cr_i2c_prd_d_ph_2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_i2c_prd_d_ph_1								cr_i2c_prd_d_ph_0							

位	名称	权限	复位值	描述
31:24	cr_i2c_prd_d_ph_3	r/w	8'd15	Length of DATA phase 3
23:16	cr_i2c_prd_d_ph_2	r/w	8'd15	Length of DATA phase 2
15:8	cr_i2c_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 Note: This value should not be set to 8'd0, adjust source clock rate instead if higher I2C clock rate is required

位	名称	权限	复位值	描述
7:0	cr_i2c_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0

## 17.9.8 i2c\_fifo\_config\_0

地址: 0x30003080



位	名称	权限	复位值	描述
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	i2c_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	i2c_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

### 17.9.9 i2c\_fifo\_config\_1

地址: 0x30003084

RSVD	rx_fifo_th	RSVD	tx_fifo_th												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD rx\_fifo\_th RSVD RSVD RSVD RSVD RSVD RSVD RSVD tx\_fifo\_th

RSVD RSVD RSVD RSVD RSVD RSVD tx\_fifo\_cnt RSVD RSVD RSVD RSVD RSVD RSVD tx\_fifo\_cnt

位	名称	权限	复位值	描述
31:25	RSVD			
24	rx_fifo_th	r/w	1'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:17	RSVD			
16	tx_fifo_th	r/w	1'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:10	RSVD			
9:8	rx_fifo_cnt	r	2'd0	RX FIFO available count
7:2	RSVD			
1:0	tx_fifo_cnt	r	2'd2	TX FIFO available count

### 17.9.10 i2c\_fifo\_wdata

地址: 0x30003088

i2c\_fifo\_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2c\_fifo\_wdata

位	名称	权限	复位值	描述
31:0	i2c_fifo_wdata	w	x	

### 17.9.11 i2c\_fifo\_rdata

地址: 0x3000308c

i2c\_fifo\_rdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2c\_fifo\_rdata

位	名称	权限	复位值	描述
31:0	i2c_fifo_rdata	r	32'h0	

# 18

## PWM

### 18.1 简介

脉冲宽度调制（Pulse width modulation，简称 PWM）是一种利用微处理器的数字信号对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。

### 18.2 主要特征

- 支持 2 组 PWM 信号生成，每组包含 4 通道 PWM 信号输出，每通道可以设置为 2 路互补 PWM
- 三种时钟源可选择（总线时钟 `<bclk>`、晶振时钟 `<xtal_ck>`、慢速时钟 `<32k>`），搭配 16-bit 时钟分频器
- 每组 PWM 都可以独立设置为不同的周期
- 每通道 PWM 都有双门限值设定，可以设定不同的占空比和相位，增加脉冲弹性
- 每通道 PWM 都有独立的死区时间设定
- 每路 PWM 输出引脚都可以设定不同的有效电平
- 每路 PWM 都有独立的连接开关用来选择是否与内部计数器相连，并可设定不连接时的默认输出电平
- 软件刹车和外部刹车信号可以将 PWM 输出电平置于预先设定的状态
- 多达 9 种可用于触发 ADC 转换的触发源
- 如下事件发生时可产生中断：计数器溢出、门限值比较匹配、刹车信号产生、PWM 周期数达到设定值

### 18.3 功能描述

#### 18.3.1 时钟与分频器

每个 PWM 计数器时钟来源都有三种选择，来源如下：

- `bclk` - 芯片的总线时钟
- `XTAL` - 外部晶振时钟
- `f32k` - 系统 RTC 时钟

每个计数器都有各自的 16-bit 分频器，可通过 APB 将选择到的时钟进行分频，PWM 计数器将以分频后的时钟作为计数周期单位，每经过一个计数周期进行数一的动作。

### 18.3.2 有效电平

不同的应用场景所需要的有效电平状态是不一样的，一部分是低电平有效一部分是高电平有效，每路 PWM 输出的有效电平可独立设定。

当 `pwm_mcx_config1` 寄存器中的位 `<pwm_chy_ppl>` 设置为 1 时表示 `pwmx` ( $x=0,1$ ) 的通道  $y$  ( $y=0,1,2,3$ ) 的正向通道被设置成高电平有效，即 PWM 模块输出逻辑 1 时其对应的引脚输出高电平，输出逻辑 0 时其对应的引脚输出低电平。如果 `<pwm_chy_ppl>` 设置为 0 则表示低电平有效，即 PWM 模块输出逻辑 1 时其对应的引脚输出低电平，输出逻辑 0 时其对应的引脚输出高电平。互补通道的设置位为 `<pwm_chy_npl>`，其设置规则与正向通道相同。

### 18.3.3 脉冲产生原理

当 `pwm_mcx_config1` 寄存器中的位 `<pwm_chy_pen>` 设置为 0 时表示 `pwmx` ( $x=0,1$ ) 的通道  $y$  ( $y=0,1,2,3$ ) 的正向通道输出被设置为一个确定的逻辑状态，此时状态由位 `<pwm_chy_psi>` 确定，当 `<pwm_chy_psi>` 为 0 时对应的正向通道引脚输出逻辑 0，当 `<pwm_chy_psi>` 为 1 时对应的正向通道引脚输出逻辑 1，实际输出电平需要和 `<pwm_chy_ppl>` 共同确定。互补通道的设置位为 `<pwm_chy_nen>`，其设置规则与正向通道相同。

当 `pwm_mcx_config1` 寄存器中的位 `<pwm_chy_pen>` (`<pwm_chy_nen>`) 设置为 1 时表示 `pwmx` ( $x=0,1$ ) 的通道  $y$  ( $y=0,1,2,3$ ) 的正向（互补）通道输出由内部计数器与两个阈值的比较结果确定，当计数器的值在两个阈值中间时，正向通道输出逻辑 1，互补通道输出逻辑 0，当计数器的值在两个阈值之外时，正向通道输出逻辑 0，互补通道输出逻辑 1。具体的波形如下图所示。

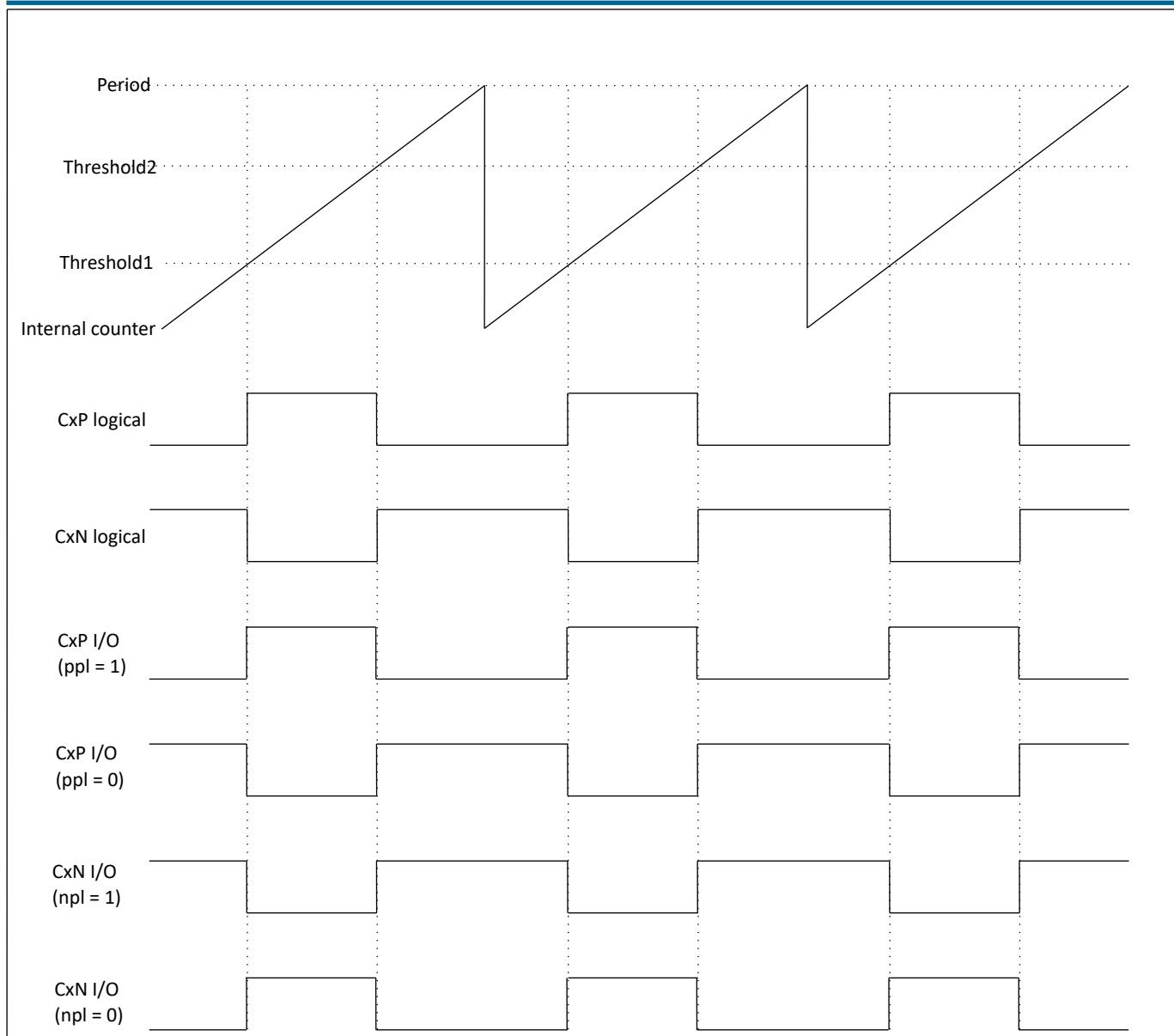


图 18.1: PWM 在不同配置下的波形图

#### 18.3.4 刹车

当使用刹车功能且刹车信号产生时，PWM 输出的电平都会被修改为预设定的状态，预设定状态由 `pwm_mcx_config1 (x=0,1)` 寄存器中的位 `<pwm_chy_pbs>`、`<pwm_chy_nbs>` ( $y=0,1,2,3$ ) 设定，当该位设置为 1 时，刹车信号产生后该路 PWM 输出逻辑 1，当该位设置为 0 时，刹车信号产生后该路 PWM 输出逻辑 0。

刹车信号包括外部刹车和软件刹车两种。当 `pwm_mcx_config0 (x=0,1)` 的位 `<pwm_sw_break_en>` 为 1 时即产生内部刹车信号，该位为 0 时即退出刹车状态，PWM 恢复之前的运行方式。当位 `<pwm_ext_break_en>` 为 1 时即使能外部刹车功能，如果此时外部刹车引脚状态与 `<pwm_ext_break_pl>` 设置的值相匹配，则产生刹车信号，如果刹车引脚状态变为与 `<pwm_ext_break_pl>` 设置的值相反则 PWM 退出刹车状态。需要注意的是只有 PWM0 有外部刹车功能，PWM1 不具备外部刹车的特性。

如果使能了刹车中断，则在外部刹车信号产生时，会触发中断。软件刹车不会触发中断。

### 18.3.5 死区

PWM 每个通道都可以独立设置不同的死区时间，当死区设定值不为 0 时，PWM 的正向通道在与阈值 1 匹配时会延迟产生跳变的电平，在阈值 2 匹配时立即改变电平状态。PWM 的互补通道在与阈值 1 匹配时会立即改变电平状态，在阈值 2 匹配时会延迟产生跳变的电平。死区长度由 `pwm_mcx_dead_time` 寄存器设置。

### 18.3.6 周期与占空比计算

PWM 的周期由两部分决定，一个是时钟分频系数，一个是时钟持续周期。时钟分频系数由寄存器 `PWMx_CLK_DIV[15:0]`( $x$  为 0~1) 进行设置，用于对 PWM 的源时钟进行分频。时钟持续周期由寄存器 `PWMx_PERIOD[15:0]`( $x$  为 0~1) 进行设置，用于设置 PWM 的一个周期由多少个分频后的时钟周期组成。即 PWM 的周期 = PWM 源时钟/`PWMx_CLK_DIV[15:0]`/`PWMx_PERIOD[15:0]`。

### 18.3.7 PWM 中断

对于每一组 PWM，可以由 `pwm_mcx_period` 寄存器的高 16 位设置周期计数值，当 PWM 的周期数达到这个计数值时，将产生 PWM 中断。

当 PWM 计数值达到周期数或计数值与阈值匹配都将产生 PWM 中断。

有外部刹车信号产生时会产生 PWM 中断。

### 18.3.8 ADC 联动

当计数器与门限值发生匹配或者计数值达到周期数时都可以产生内部触发 ADC 启动转换的信号，需要注意的是只有 PWM0 可以触发 ADC 启动转换，PWM1 不具备该特性。具体的触发源由 `pwm_mcx_config0` 寄存器的位 `<pwm_adc_trg_src>` 设置。这种特性在定时采样的应用场景中较为常用。以下举例说明。

应用场景：在 BLDC 的应用中，有这样一种应用需求，在用 PWM 控制电机的转速的同时，还需要检测流过线圈的电流。在一个 PWM 周期内，PWM 控制功率器件导通后，经过一段特定的时间电流达到稳定，此时需要采样电流值，这意味着触发 ADC 转换的时刻点与 PWM 之间有严格的相位差。比如 PWM 的通道 0 用于驱动电机的其中一相，需要产生 10KHz 占空比为 20% 的方波，并且需要在方波高电平的中间时刻点执行 ADC 采样，则该 PWM 周期为 100us，当时钟源为 1MHz 时，周期计数值为 100，可以将通道 1 的两个阈值分别设置为 0 和 20，此时计数器在 0~20 之间时，PWM 输出高电平，否则输出低电平。将通道 2 的阈值 L 设置为 10，且将 `pwm_mc0_config0` 寄存器中的 `pwm_adc_trg_src` 设置为 4 即 `pwm_ch2l_int` 可以触发 ADC 转换，则计数器数到 10 时即通道 1 产生的高电平中间时刻会启动 ADC 开始采样转换，这样就可以保证每次采样都满足精确的时间要求，且不需要 CPU 干预，从而提高了性能。

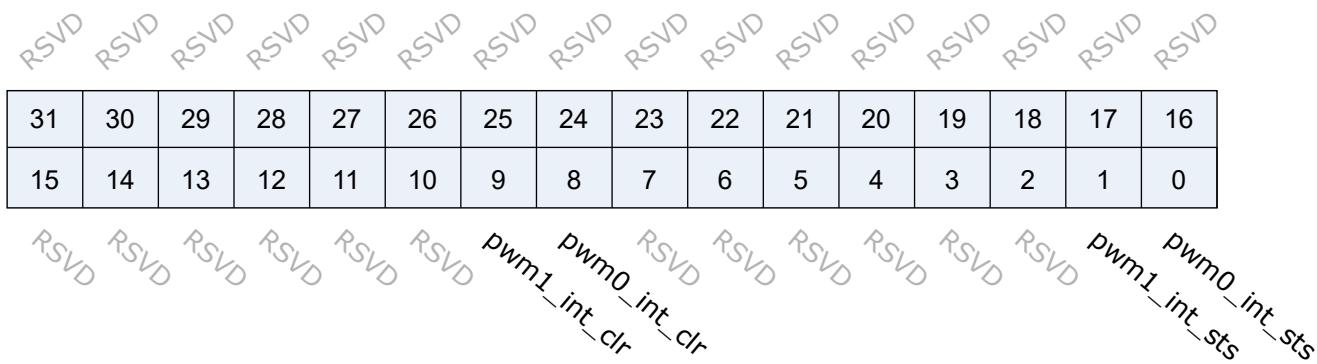
## 18.4 寄存器描述

名称	描述
<code>pwm_int_config</code>	
<code>pwm_mc0_config0</code>	
<code>pwm_mc0_config1</code>	
<code>pwm_mc0_period</code>	
<code>pwm_mc0_dead_time</code>	

名称	描述
pwm_mc0_ch0_thre	
pwm_mc0_ch1_thre	
pwm_mc0_ch2_thre	
pwm_mc0_ch3_thre	
pwm_mc0_int_sts	
pwm_mc0_int_mask	
pwm_mc0_int_clear	
pwm_mc0_int_en	
pwm_mc1_config0	
pwm_mc1_config1	
pwm_mc1_period	
pwm_mc1_dead_time	
pwm_mc1_ch0_thre	
pwm_mc1_ch1_thre	
pwm_mc1_ch2_thre	
pwm_mc1_ch3_thre	
pwm_mc1_int_sts	
pwm_mc1_int_mask	
pwm_mc1_int_clear	
pwm_mc1_int_en	

#### 18.4.1 pwm\_int\_config

地址: 0x2000a400



位	名称	权限	复位值	描述
31:10	RSVD			
9	pwm1_int_clr	w1c	1'b0	PWM 1 interrupt clear (clear pwm_mc1_int_sts[10:0] all)
8	pwm0_int_clr	w1c	1'b0	PWM 0 interrupt clear (clear pwm_mc0_int_sts[10:0] all)
7:2	RSVD			
1	pwm1_int_sts	r	1'b0	PWM 1 interrupt status (Check pwm_mc1_int_sts for detailed interrupt status)
0	pwm0_int_sts	r	1'b0	PWM 0 interrupt status (Check pwm_mc0_int_sts for detailed interrupt status)

#### 18.4.2 pwm\_mc0\_config0

地址: 0x2000a440

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pwm_clk_div															

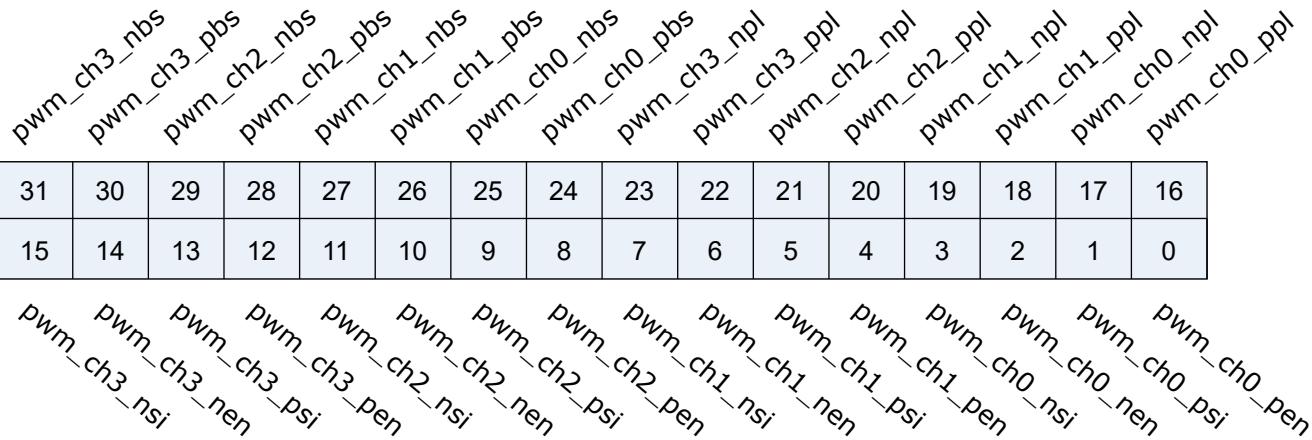
reg\_clk\_sel  
 pwm\_sts\_stop  
 pwm\_stop\_mode  
 pwm\_stop\_en  
 pwm\_ext\_break\_pl  
 pwm\_ext\_break\_en  
 pwm\_sw\_break\_en  
 pwm\_adc\_trg\_src  
 pwm\_stop\_on\_rept  
 RSVD  
 RSVD  
 RSVD

位	名称	权限	复位值	描述
31:30	reg_clk_sel	r/w	2'd0	PWM clock source select, 2'b00-xclk ; 2'b01-bclk ; others-f32k_clk
29	pwm_sts_stop	r	1'b0	PWM stop status
28	pwm_stop_mode	r/w	1'b1	PWM stop mode, 1'b1 - graceful ; 1'b0 - abrupt
27	pwm_stop_en	r/w	1'b0	PWM stop enable
26	pwm_ext_break_pl	r/w	1'b0	PWM external break source polarity 1'b0: Active-LOW 1'b1: Active-HIGH

位	名称	权限	复位值	描述
25	pwm_ext_break_en	r/w	1'b0	PWM external break source enable 1'b0: Disabled, external break signal is masked 1'b1: Enabled, external break signal is effective
24	pwm_sw_break_en	r/w	1'b0	PWM break enable 1'b0: Disabled, normal operation 1'b1: Enabled, PWM output will be determined by CxPBS/CxNBS
23:20	pwm_adc_trg_src	r/w	4'hF	Select signal of ADC triggering source 4'd0: pwm_ch0l_int (Channel 0 ThresholdL reached) 4'd1: pwm_ch0h_int (Channel 0 ThresholdH reached) 4'd2: pwm_ch1l_int (Channel 1 ThresholdL reached) 4'd3: pwm_ch1h_int (Channel 1 ThresholdH reached) 4'd4: pwm_ch2l_int (Channel 2 ThresholdL reached) 4'd5: pwm_ch2h_int (Channel 2 ThresholdH reached) 4'd6: pwm_ch3l_int (Channel 3 ThresholdL reached) 4'd7: pwm_ch3h_int (Channel 3 ThresholdH reached) 4'd8: pwm_prde_int (Period End reached) Others: Disabled
19	pwm_stop_on_rept	r/w	1'b0	PWM stopped when rept_int is asserted 1'b0: Disabled, PWM keeps running when rept_int is asserted 1'b1: Enabled, PWM stops when rept_int is asserted. Clear rept_int to restore operation
18:16	RSVD			
15:0	pwm_clk_div	r/w	16'b0	PWM clock division

### 18.4.3 pwm\_mc0\_config1

地址: 0x2000a444



位	名称	权限	复位值	描述
31	pwm_ch3_nbs	r/w	1'b0	PWM channel 3 negative break state
30	pwm_ch3_pbs	r/w	1'b0	PWM channel 3 positive break state
29	pwm_ch2_nbs	r/w	1'b0	PWM channel 2 negative break state
28	pwm_ch2_pbs	r/w	1'b0	PWM channel 2 positive break state
27	pwm_ch1_nbs	r/w	1'b0	PWM channel 1 negative break state
26	pwm_ch1_pbs	r/w	1'b0	PWM channel 1 positive break state
25	pwm_ch0_nbs	r/w	1'b0	PWM channel 0 negative break state
24	pwm_ch0_pbs	r/w	1'b0	PWM channel 0 positive break state
23	pwm_ch3_npl	r/w	1'b1	PWM channel 3 negative polarity
22	pwm_ch3_ppl	r/w	1'b1	PWM channel 3 positive polarity
21	pwm_ch2_npl	r/w	1'b1	PWM channel 2 negative polarity
20	pwm_ch2_ppl	r/w	1'b1	PWM channel 2 positive polarity
19	pwm_ch1_npl	r/w	1'b1	PWM channel 1 negative polarity
18	pwm_ch1_ppl	r/w	1'b1	PWM channel 1 positive polarity
17	pwm_ch0_npl	r/w	1'b1	PWM channel 0 negative polarity
16	pwm_ch0_ppl	r/w	1'b1	PWM channel 0 positive polarity
15	pwm_ch3_nsi	r/w	1'b1	PWM channel 3 negative set idle state
14	pwm_ch3_nen	r/w	1'b0	PWM channel 3 negative enable pwm out
13	pwm_ch3_psi	r/w	1'b0	PWM channel 3 positive set idle state
12	pwm_ch3_pen	r/w	1'b0	PWM channel 3 positive enable pwm out

位	名称	权限	复位值	描述
11	pwm_ch2_nsi	r/w	1'b1	PWM channel 2 negative set idle state
10	pwm_ch2_nen	r/w	1'b0	PWM channel 2 negative enable pwm out
9	pwm_ch2_psi	r/w	1'b0	PWM channel 2 positive set idle state
8	pwm_ch2_pen	r/w	1'b0	PWM channel 2 positive enable pwm out
7	pwm_ch1_nsi	r/w	1'b1	PWM channel 1 negative set idle state
6	pwm_ch1_nen	r/w	1'b0	PWM channel 1 negative enable pwm out
5	pwm_ch1_psi	r/w	1'b0	PWM channel 1 positive set idle state
4	pwm_ch1_pen	r/w	1'b0	PWM channel 1 positive enable pwm out
3	pwm_ch0_nsi	r/w	1'b1	PWM channel 0 negative set idle state
2	pwm_ch0_nen	r/w	1'b0	PWM channel 0 negative enable pwm out
1	pwm_ch0_psi	r/w	1'b0	PWM channel 0 positive set idle state
0	pwm_ch0_pen	r/w	1'b0	PWM channel 0 positive enable pwm out

#### 18.4.4 pwm\_mc0\_period

地址: 0x2000a448

pwm\_int\_period\_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_period

位	名称	权限	复位值	描述
31:16	pwm_int_period_cnt	r/w	16'd0	PWM interrupt period counter threshold
15:0	pwm_period	r/w	16'd0	PWM period setting

#### 18.4.5 pwm\_mc0\_dead\_time

地址: 0x2000a44c

pwm\_ch3\_dtg

pwm\_ch2\_dtg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch1\_dtg

pwm\_ch0\_dtg

位	名称	权限	复位值	描述
31:24	pwm_ch3_dtg	r/w	8'h0	PWM Channel 3 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
23:16	pwm_ch2_dtg	r/w	8'h0	PWM Channel 2 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
15:8	pwm_ch1_dtg	r/w	8'h0	PWM Channel 1 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
7:0	pwm_ch0_dtg	r/w	8'h0	PWM Channel 0 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

### 18.4.6 pwm\_mc0\_ch0\_thre

地址: 0x2000a450

pwm\_ch0\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch0\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch0_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch0_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

### 18.4.7 pwm\_mc0\_ch1\_thre

地址: 0x2000a454

pwm\_ch1\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch1\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch1_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch1_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

### 18.4.8 pwm\_mc0\_ch2\_thre

地址: 0x2000a458

pwm\_ch2\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch2\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch2_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch2_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

#### 18.4.9 pwm\_mc0\_ch3\_thre

地址: 0x2000a45c

pwm\_ch3\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch3\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch3_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch3_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

#### 18.4.10 pwm\_mc0\_int\_sts

地址: 0x2000a460

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD

位	名称	权限	复位值	描述
31:11	RSVD			
10	pwm_rept_int	r	1'b0	PWM repeat count interrupt status

位	名称	权限	复位值	描述
9	pwm_brk_int	r	1'b0	PWM break interrupt status, triggered when ext_break is asserted Note: pwm_sw_break_en will NOT trigger this interrupt
8	pwm_prde_int	r	1'b0	PWM period end interrupt status
7	pwm_ch3h_int	r	1'b0	PWM Channel 3 ThresholdH interrupt status
6	pwm_ch3l_int	r	1'b0	PWM Channel 3 ThresholdL interrupt status
5	pwm_ch2h_int	r	1'b0	PWM Channel 2 ThresholdH interrupt status
4	pwm_ch2l_int	r	1'b0	PWM Channel 2 ThresholdL interrupt status
3	pwm_ch1h_int	r	1'b0	PWM Channel 1 ThresholdH interrupt status
2	pwm_ch1l_int	r	1'b0	PWM Channel 1 ThresholdL interrupt status
1	pwm_ch0h_int	r	1'b0	PWM Channel 0 ThresholdH interrupt status
0	pwm_ch0l_int	r	1'b0	PWM Channel 0 ThresholdL interrupt status

#### 18.4.11 pwm\_mc0\_int\_mask

地址: 0x2000a464

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
cr_pwm_rept_mask cr_pwm_brk_mask cr_pwm_prde_mask cr_pwm_ch3h_mask cr_pwm_ch3l_mask cr_pwm_ch2h_mask cr_pwm_ch2l_mask cr_pwm_ch1h_mask cr_pwm_ch1l_mask cr_pwm_ch0h_mask cr_pwm_ch0l_mask																

位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_pwm_rept_mask	r/w	1'b1	Interrupt mask of pwm_rept_int
9	cr_pwm_brk_mask	r/w	1'b1	Interrupt mask of pwm_brk_int
8	cr_pwm_prde_mask	r/w	1'b1	Interrupt mask of pwm_prde_int
7	cr_pwm_ch3h_mask	r/w	1'b1	Interrupt mask of pwm_ch3h_int
6	cr_pwm_ch3l_mask	r/w	1'b1	Interrupt mask of pwm_ch3l_int

位	名称	权限	复位值	描述
5	cr_pwm_ch2h_mask	r/w	1'b1	Interrupt mask of pwm_ch2h_int
4	cr_pwm_ch2l_mask	r/w	1'b1	Interrupt mask of pwm_ch2l_int
3	cr_pwm_ch1h_mask	r/w	1'b1	Interrupt mask of pwm_ch1h_int
2	cr_pwm_ch1l_mask	r/w	1'b1	Interrupt mask of pwm_ch1l_int
1	cr_pwm_ch0h_mask	r/w	1'b1	Interrupt mask of pwm_ch0h_int
0	cr_pwm_ch0l_mask	r/w	1'b1	Interrupt mask of pwm_ch0l_int

#### 18.4.12 pwm\_mc0\_int\_clear

地址: 0x2000a468

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RSVD	RSVD	RSVD	RSVD	RSVD	cr_pwm_rept_clr	cr_pwm_brk_clr	cr_pwm_prde_clr	cr_pwm_ch3h_clr	cr_pwm_ch3l_clr	cr_pwm_ch2h_clr	cr_pwm_ch2l_clr	cr_pwm_ch1h_clr	cr_pwm_ch1l_clr	cr_pwm_ch0h_clr	cr_pwm_ch0l_clr	

位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_pwm_rept_clr	w1c	1'b0	Interrupt clear of pwm_rept_int
9	cr_pwm_brk_clr	w1c	1'b0	Interrupt clear of pwm_brk_int
8	cr_pwm_prde_clr	w1c	1'b0	Interrupt clear of pwm_prde_int
7	cr_pwm_ch3h_clr	w1c	1'b0	Interrupt clear of pwm_ch3h_int
6	cr_pwm_ch3l_clr	w1c	1'b0	Interrupt clear of pwm_ch3l_int
5	cr_pwm_ch2h_clr	w1c	1'b0	Interrupt clear of pwm_ch2h_int
4	cr_pwm_ch2l_clr	w1c	1'b0	Interrupt clear of pwm_ch2l_int
3	cr_pwm_ch1h_clr	w1c	1'b0	Interrupt clear of pwm_ch1h_int
2	cr_pwm_ch1l_clr	w1c	1'b0	Interrupt clear of pwm_ch1l_int
1	cr_pwm_ch0h_clr	w1c	1'b0	Interrupt clear of pwm_ch0h_int
0	cr_pwm_ch0l_clr	w1c	1'b0	Interrupt clear of pwm_ch0l_int

### 18.4.13 pwm\_mc0\_int\_en

地址: 0x2000a46c

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD cr\_pwm\_rept\_en cr\_pwm\_brk\_en cr\_pwm\_prde\_en cr\_pwm\_ch3h\_en cr\_pwm\_ch3l\_en cr\_pwm\_ch2h\_en cr\_pwm\_ch2l\_en cr\_pwm\_ch1h\_en cr\_pwm\_ch1l\_en cr\_pwm\_ch0h\_en cr\_pwm\_ch0l\_en

位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_pwm_rept_en	r/w	1'b1	Interrupt enable of pwm_rept_int
9	cr_pwm_brk_en	r/w	1'b1	Interrupt enable of pwm_brk_int
8	cr_pwm_prde_en	r/w	1'b1	Interrupt enable of pwm_prde_int
7	cr_pwm_ch3h_en	r/w	1'b1	Interrupt enable of pwm_ch3h_int
6	cr_pwm_ch3l_en	r/w	1'b1	Interrupt enable of pwm_ch3l_int
5	cr_pwm_ch2h_en	r/w	1'b1	Interrupt enable of pwm_ch2h_int
4	cr_pwm_ch2l_en	r/w	1'b1	Interrupt enable of pwm_ch2l_int
3	cr_pwm_ch1h_en	r/w	1'b1	Interrupt enable of pwm_ch1h_int
2	cr_pwm_ch1l_en	r/w	1'b1	Interrupt enable of pwm_ch1l_int
1	cr_pwm_ch0h_en	r/w	1'b1	Interrupt enable of pwm_ch0h_int
0	cr_pwm_ch0l_en	r/w	1'b1	Interrupt enable of pwm_ch0l_int

### 18.4.14 pwm\_mc1\_config0

地址: 0x2000a480

reg_clk_sel	pwm_sts_stop	pwm_stop_mode	pwm_stop_en	pwm_ext_break_pl	pwm_ext_break_en	pwm_sw_break_en	pwm_adc_trg_src	pwm_stop_on_rept	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20
15	14	13	12	11	10	9	8	7	6	5	4

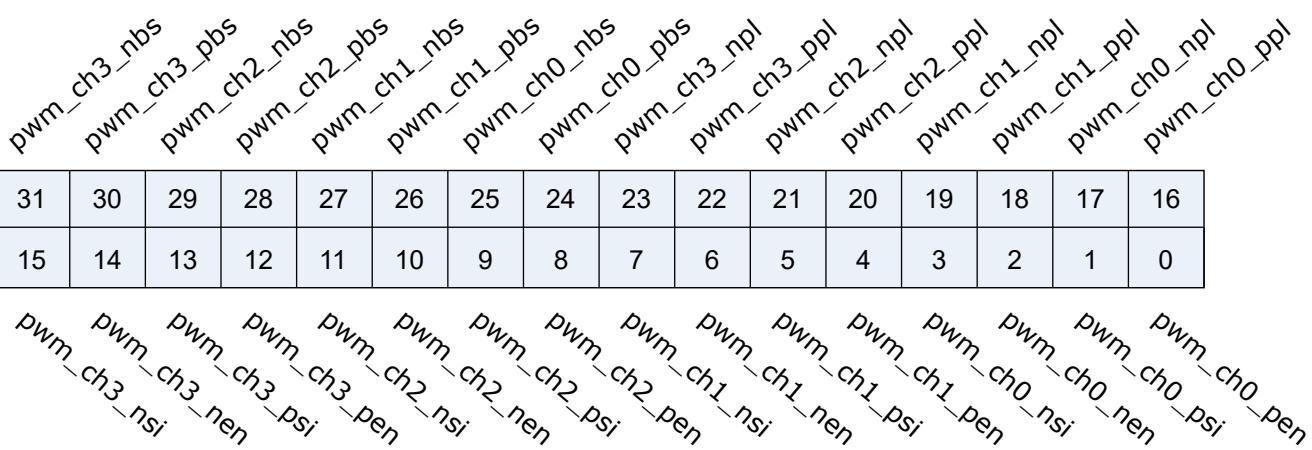
pwm\_clk\_div

位	名称	权限	复位值	描述
31:30	reg_clk_sel	r/w	2'd0	PWM clock source select, 2'b00-xclk ; 2'b01-bclk ; others-f32k_clk
29	pwm_sts_stop	r	1'b0	PWM stop status
28	pwm_stop_mode	r/w	1'b1	PWM stop mode, 1'b1 - graceful ; 1'b0 - abrupt
27	pwm_stop_en	r/w	1'b0	PWM stop enable
26	pwm_ext_break_pl	r/w	1'b0	PWM external break source polarity 1'b0: Active-LOW 1'b1: Active-HIGH
25	pwm_ext_break_en	r/w	1'b0	PWM external break source enable 1'b0: Disabled, external break signal is masked 1'b1: Enabled, external break signal is effective
24	pwm_sw_break_en	r/w	1'b0	PWM break enable 1'b0: Disabled, normal operation 1'b1: Enabled, PWM output will be determined by CxPBS/CxNBS
23:20	pwm_adc_trg_src	r/w	4'hF	Select signal of ADC triggering source 4'd0: pwm_ch0l_int (Channel 0 ThresholdL reached) 4'd1: pwm_ch0h_int (Channel 0 ThresholdH reached) 4'd2: pwm_ch1l_int (Channel 1 ThresholdL reached) 4'd3: pwm_ch1h_int (Channel 1 ThresholdH reached) 4'd4: pwm_ch2l_int (Channel 2 ThresholdL reached) 4'd5: pwm_ch2h_int (Channel 2 ThresholdH reached) 4'd6: pwm_ch3l_int (Channel 3 ThresholdL reached) 4'd7: pwm_ch3h_int (Channel 3 ThresholdH reached) 4'd8: pwm_prde_int (Period End reached) Others: Disabled

位	名称	权限	复位值	描述
19	pwm_stop_on_rept	r/w	1'b0	PWM stopped when rept_int is asserted 1'b0: Disabled, PWM keeps running when rept_int is asserted 1'b1: Enabled, PWM stops when rept_int is asserted. Clear rept_int to restore operation
18:16	RSVD			
15:0	pwm_clk_div	r/w	16'b0	PWM clock division

#### 18.4.15 pwm\_mc1\_config1

地址: 0x2000a484



位	名称	权限	复位值	描述
31	pwm_ch3_nbs	r/w	1'b0	PWM channel 3 negative break state
30	pwm_ch3_pbs	r/w	1'b0	PWM channel 3 positive break state
29	pwm_ch2_nbs	r/w	1'b0	PWM channel 2 negative break state
28	pwm_ch2_pbs	r/w	1'b0	PWM channel 2 positive break state
27	pwm_ch1_nbs	r/w	1'b0	PWM channel 1 negative break state
26	pwm_ch1_pbs	r/w	1'b0	PWM channel 1 positive break state
25	pwm_ch0_nbs	r/w	1'b0	PWM channel 0 negative break state
24	pwm_ch0_pbs	r/w	1'b0	PWM channel 0 positive break state
23	pwm_ch3_npl	r/w	1'b1	PWM channel 3 negative polarity
22	pwm_ch3_ppl	r/w	1'b1	PWM channel 3 positive polarity
21	pwm_ch2_npl	r/w	1'b1	PWM channel 2 negative polarity
20	pwm_ch2_ppl	r/w	1'b1	PWM channel 2 positive polarity

位	名称	权限	复位值	描述
19	pwm_ch1_npl	r/w	1'b1	PWM channel 1 negative polarity
18	pwm_ch1_ppl	r/w	1'b1	PWM channel 1 positive polarity
17	pwm_ch0_npl	r/w	1'b1	PWM channel 0 negative polarity
16	pwm_ch0_ppl	r/w	1'b1	PWM channel 0 positive polarity
15	pwm_ch3_nsi	r/w	1'b1	PWM channel 3 negative set idle state
14	pwm_ch3_nen	r/w	1'b0	PWM channel 3 negative enable pwm out
13	pwm_ch3_psi	r/w	1'b0	PWM channel 3 positive set idle state
12	pwm_ch3_pen	r/w	1'b0	PWM channel 3 positive enable pwm out
11	pwm_ch2_nsi	r/w	1'b1	PWM channel 2 negative set idle state
10	pwm_ch2_nen	r/w	1'b0	PWM channel 2 negative enable pwm out
9	pwm_ch2_psi	r/w	1'b0	PWM channel 2 positive set idle state
8	pwm_ch2_pen	r/w	1'b0	PWM channel 2 positive enable pwm out
7	pwm_ch1_nsi	r/w	1'b1	PWM channel 1 negative set idle state
6	pwm_ch1_nen	r/w	1'b0	PWM channel 1 negative enable pwm out
5	pwm_ch1_psi	r/w	1'b0	PWM channel 1 positive set idle state
4	pwm_ch1_pen	r/w	1'b0	PWM channel 1 positive enable pwm out
3	pwm_ch0_nsi	r/w	1'b1	PWM channel 0 negative set idle state
2	pwm_ch0_nen	r/w	1'b0	PWM channel 0 negative enable pwm out
1	pwm_ch0_psi	r/w	1'b0	PWM channel 0 positive set idle state
0	pwm_ch0_pen	r/w	1'b0	PWM channel 0 positive enable pwm out

#### 18.4.16 pwm\_mc1\_period

地址: 0x2000a488

pwm\_int\_period\_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_period

位	名称	权限	复位值	描述
31:16	pwm_int_period_cnt	r/w	16'd0	PWM interrupt period counter threshold
15:0	pwm_period	r/w	16'd0	PWM period setting

### 18.4.17 pwm\_mc1\_dead\_time

地址: 0x2000a48c

pwm_ch3_dtg								pwm_ch2_dtg							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pwm_ch1_dtg								pwm_ch0_dtg							

位	名称	权限	复位值	描述
31:24	pwm_ch3_dtg	r/w	8'h0	PWM Channel 3 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
23:16	pwm_ch2_dtg	r/w	8'h0	PWM Channel 2 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
15:8	pwm_ch1_dtg	r/w	8'h0	PWM Channel 1 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

位	名称	权限	复位值	描述
7:0	pwm_ch0_dtg	r/w	8'h0	PWM Channel 0 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

#### 18.4.18 pwm\_mc1\_ch0\_thre

地址: 0x2000a490

pwm\_ch0\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch0\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch0_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch0_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

#### 18.4.19 pwm\_mc1\_ch1\_thre

地址: 0x2000a494

pwm\_ch1\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch1\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch1_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL

位	名称	权限	复位值	描述
15:0	pwm_ch1_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

#### 18.4.20 pwm\_mc1\_ch2\_thre

地址: 0x2000a498

pwm\_ch2\_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch2\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch2_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch2_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

#### 18.4.21 pwm\_mc1\_ch3\_thre

地址: 0x2000a49c

pwm\_ch3\_threH

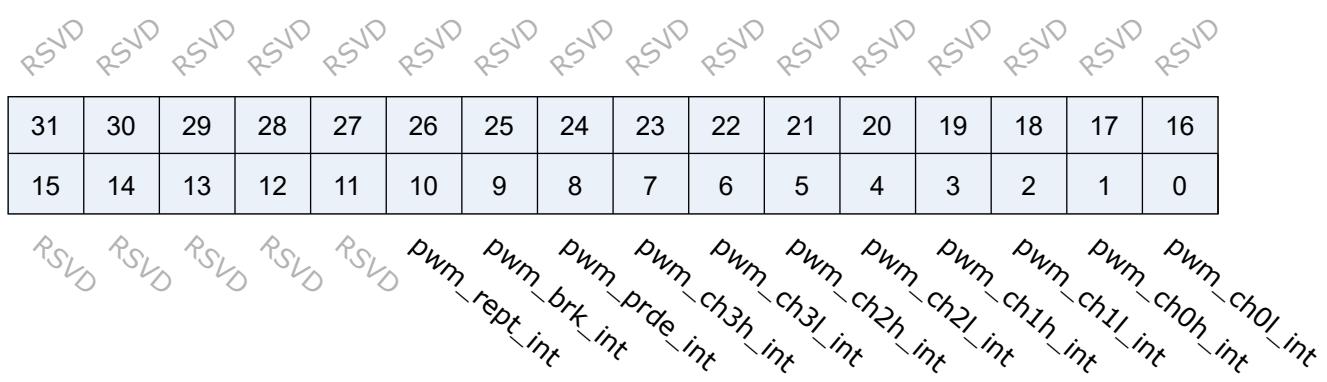
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm\_ch3\_threL

位	名称	权限	复位值	描述
31:16	pwm_ch3_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch3_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

### 18.4.22 pwm\_mc1\_int\_sts

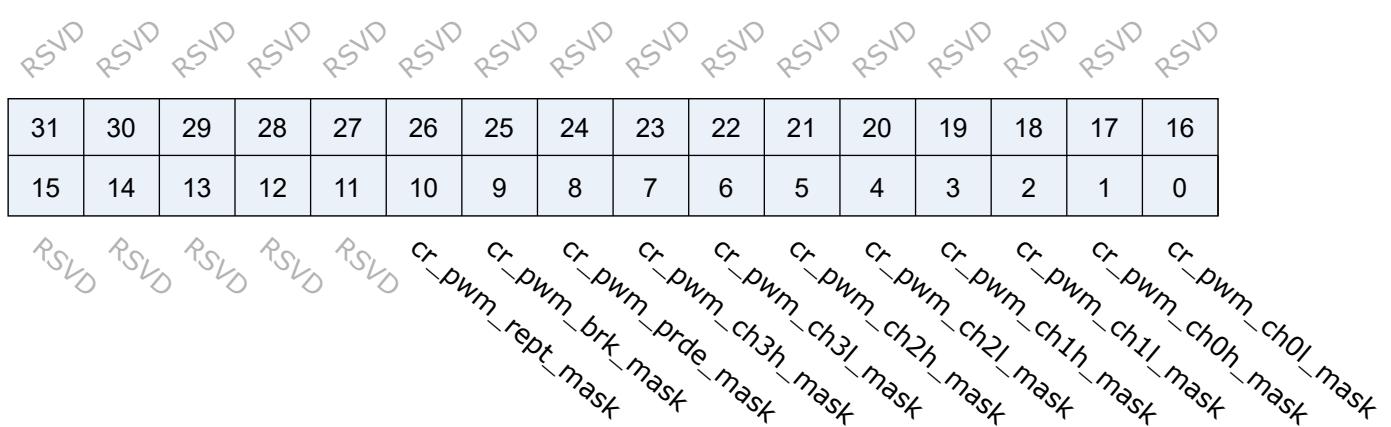
地址: 0x2000a4a0



位	名称	权限	复位值	描述
31:11	RSVD			
10	pwm_rept_int	r	1'b0	PWM repeat count interrupt status
9	pwm_brk_int	r	1'b0	PWM break interrupt status, triggered when ext_break is asserted Note: pwm_sw_break_en will NOT trigger this interrupt
8	pwm_prde_int	r	1'b0	PWM period end interrupt status
7	pwm_ch3h_int	r	1'b0	PWM Channel 3 ThresholdH interrupt status
6	pwm_ch3l_int	r	1'b0	PWM Channel 3 ThresholdL interrupt status
5	pwm_ch2h_int	r	1'b0	PWM Channel 2 ThresholdH interrupt status
4	pwm_ch2l_int	r	1'b0	PWM Channel 2 ThresholdL interrupt status
3	pwm_ch1h_int	r	1'b0	PWM Channel 1 ThresholdH interrupt status
2	pwm_ch1l_int	r	1'b0	PWM Channel 1 ThresholdL interrupt status
1	pwm_ch0h_int	r	1'b0	PWM Channel 0 ThresholdH interrupt status
0	pwm_ch0l_int	r	1'b0	PWM Channel 0 ThresholdL interrupt status

### 18.4.23 pwm\_mc1\_int\_mask

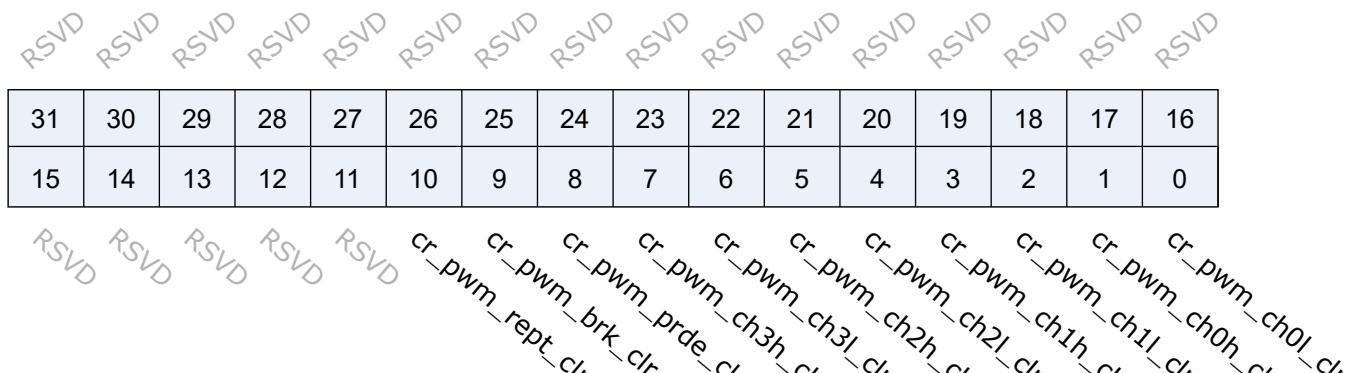
地址: 0x2000a4a4



位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_pwm_rept_mask	r/w	1'b1	Interrupt mask of pwm_rept_int
9	cr_pwm_brk_mask	r/w	1'b1	Interrupt mask of pwm_brk_int
8	cr_pwm_prde_mask	r/w	1'b1	Interrupt mask of pwm_prde_int
7	cr_pwm_ch3h_mask	r/w	1'b1	Interrupt mask of pwm_ch3h_int
6	cr_pwm_ch3l_mask	r/w	1'b1	Interrupt mask of pwm_ch3l_int
5	cr_pwm_ch2h_mask	r/w	1'b1	Interrupt mask of pwm_ch2h_int
4	cr_pwm_ch2l_mask	r/w	1'b1	Interrupt mask of pwm_ch2l_int
3	cr_pwm_ch1h_mask	r/w	1'b1	Interrupt mask of pwm_ch1h_int
2	cr_pwm_ch1l_mask	r/w	1'b1	Interrupt mask of pwm_ch1l_int
1	cr_pwm_ch0h_mask	r/w	1'b1	Interrupt mask of pwm_ch0h_int
0	cr_pwm_ch0l_mask	r/w	1'b1	Interrupt mask of pwm_ch0l_int

### 18.4.24 pwm\_mc1\_int\_clear

地址: 0x2000a4a8



位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_pwm_rept_clr	w1c	1'b0	Interrupt clear of pwm_rept_int
9	cr_pwm_brk_clr	w1c	1'b0	Interrupt clear of pwm_brk_int
8	cr_pwm_prde_clr	w1c	1'b0	Interrupt clear of pwm_prde_int
7	cr_pwm_ch3h_clr	w1c	1'b0	Interrupt clear of pwm_ch3h_int
6	cr_pwm_ch3l_clr	w1c	1'b0	Interrupt clear of pwm_ch3l_int
5	cr_pwm_ch2h_clr	w1c	1'b0	Interrupt clear of pwm_ch2h_int
4	cr_pwm_ch2l_clr	w1c	1'b0	Interrupt clear of pwm_ch2l_int
3	cr_pwm_ch1h_clr	w1c	1'b0	Interrupt clear of pwm_ch1h_int
2	cr_pwm_ch1l_clr	w1c	1'b0	Interrupt clear of pwm_ch1l_int
1	cr_pwm_ch0h_clr	w1c	1'b0	Interrupt clear of pwm_ch0h_int
0	cr_pwm_ch0l_clr	w1c	1'b0	Interrupt clear of pwm_ch0l_int

### 18.4.25 pwm\_mc1\_int\_en

地址: 0x2000a4ac

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD      RSVD      RSVD      RSVD      RSVD  
 cr\_pwm\_rept\_en      cr\_pwm\_brk\_en      cr\_pwm\_prde\_en      cr\_pwm\_ch3h\_en      cr\_pwm\_ch3l\_en      cr\_pwm\_ch2h\_en      cr\_pwm\_ch2l\_en      cr\_pwm\_ch1h\_en      cr\_pwm\_ch1l\_en      cr\_pwm\_ch0h\_en      cr\_pwm\_ch0l\_en

位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_pwm_rept_en	r/w	1'b1	Interrupt enable of pwm_rept_int
9	cr_pwm_brk_en	r/w	1'b1	Interrupt enable of pwm_brk_int
8	cr_pwm_prde_en	r/w	1'b1	Interrupt enable of pwm_prde_int
7	cr_pwm_ch3h_en	r/w	1'b1	Interrupt enable of pwm_ch3h_int
6	cr_pwm_ch3l_en	r/w	1'b1	Interrupt enable of pwm_ch3l_int
5	cr_pwm_ch2h_en	r/w	1'b1	Interrupt enable of pwm_ch2h_int
4	cr_pwm_ch2l_en	r/w	1'b1	Interrupt enable of pwm_ch2l_int
3	cr_pwm_ch1h_en	r/w	1'b1	Interrupt enable of pwm_ch1h_int
2	cr_pwm_ch1l_en	r/w	1'b1	Interrupt enable of pwm_ch1l_int
1	cr_pwm_ch0h_en	r/w	1'b1	Interrupt enable of pwm_ch0h_int
0	cr_pwm_ch0l_en	r/w	1'b1	Interrupt enable of pwm_ch0l_int

## 19.1 简介

芯片内置 2 组 32-bit 计数器，每个计数器可独立控制配置其参数与时钟频率。

芯片内有一组看门狗计数器，不可预知的软件或硬件行为有可能导致应用程序工作失常，看门狗定时器可以帮助系统从中恢复，如果当前阶段超过预定时间，但没有喂狗或关闭看门狗定时器，可依设定触发中断或系统复位。

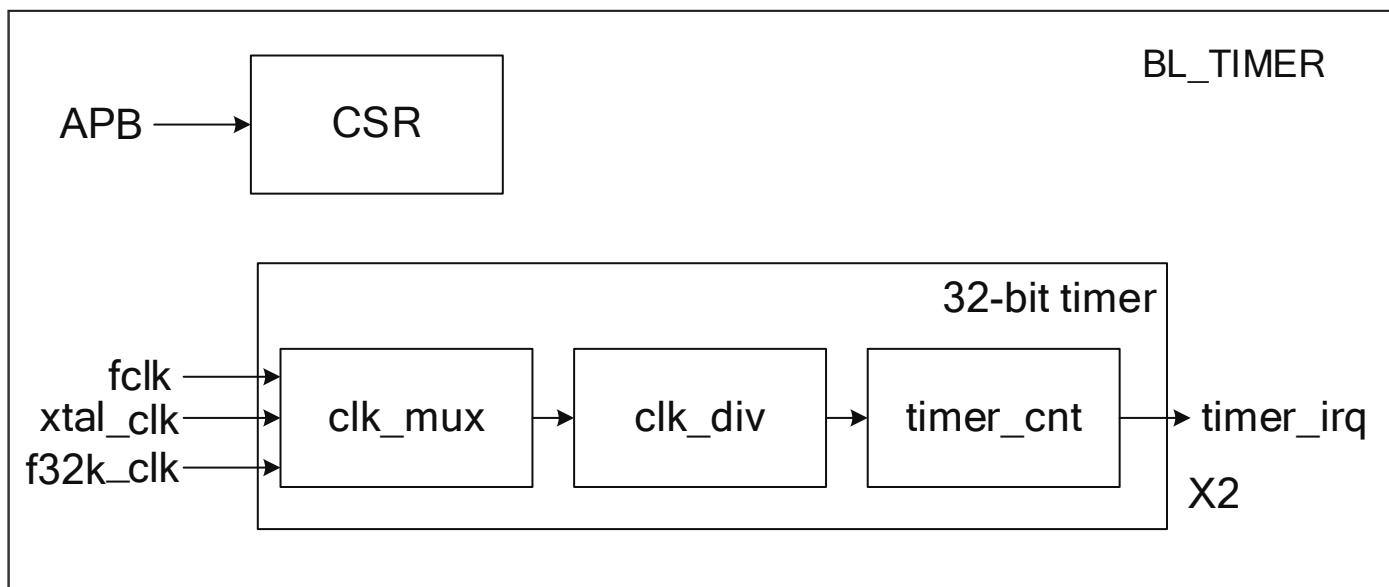


图 19.1: 定时器框图

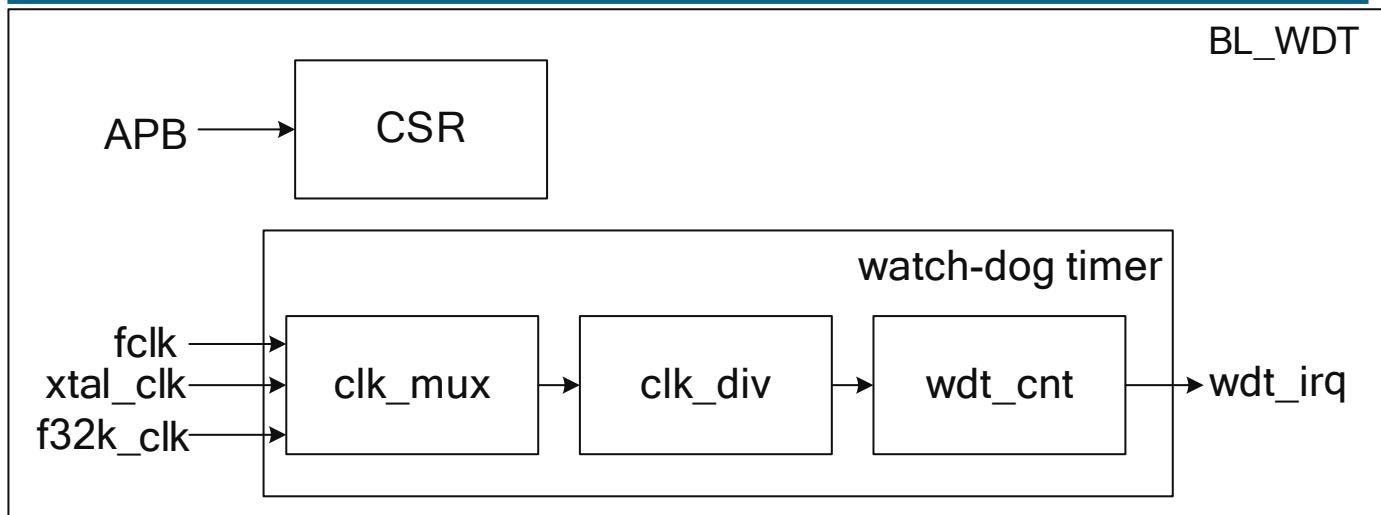


图 19.2: 看门狗定时器框图

## 19.2 主要特征

- 多种时钟来源，最高可支持 80M 时钟
- 8-bit 时钟分频器，分频系数为 1-256
- 两组 32-bit 定时器
- 每个定时器包含三组报警值设定，可独立设定每组报警值溢出时报警
- 支持 Free Run 模式和 Pre\_load 模式
- 16-bit 看门狗定时器
- 支持写入密码保护，防止误设定造成系统异常
- 支持中断或复位两种看门狗溢出方式
- 支持测量外部 GPIO 的脉冲宽度

## 19.3 功能描述

Watchdog 定时器时钟有 5 种选择：

- Bclk--总线时钟
- 32K--32K 时钟
- 1K--1K 时钟
- Xtal--外部晶振
- GPIO--外部 GPIO

通过寄存器 TCCR 中 cs\_wdt 配置。

每个定时器时钟来源都有 5 种选择，来源如下：

- Bclk--总线时钟

- 32K--32K 时钟
- 1K--1K 时钟（32K 的分频）
- Xtal--外部晶振
- GPIO--外部 GPIO

通过寄存器 TCCR 中 cs\_2 和 cs\_3 配置。

每个计数器有各自的 8-bit 分频器，可将的时钟进行 1-256 的分频，具体来说设定为 0 时表示不分频，设定为 1 时进行 2 分频以此类推，最大分频系数为 256，计数器将以分频后的时钟作为计数周期单位。

通过寄存器 TCDR 中 tcdr2、tcdr3、wcdr 配置。

### 19.3.1 通用定时器工作原理

每个通用定时器都包含三组比较器，一个计数器以及一个预加载寄存器，当设定好时钟源，启动定时器后，计数器开始向上累加计数，当计数器的值与比较器相等的时候，比较标志置位同时可以产生比较中断。

配置寄存器 TICR2 中 tclr2\_0 设置 TMR2 比较器 0 的值，配置寄存器 TICR2 中 tclr2\_1 设置 TMR2 比较器 1 的值，配置寄存器 TICR2 中 tclr2\_2 设置 TMR2 比较器 2 的值。配置寄存器 TPLVR2 中 tplvr2 设置 TMR2 预加载值。

配置寄存器 TICR3 中 tclr3\_0 设置 TMR3 比较器 0 的值，配置寄存器 TICR3 中 tclr3\_1 设置 TMR3 比较器 1 的值，配置寄存器 TICR3 中 tclr3\_2 设置 TMR3 比较器 2 的值。配置寄存器 TPLVR3 中 tplvr3 设置 TMR3 预加载值。

计数器的初始值取决于定时的模式，在 FreeRun 模式下，计数器的初始值是 0，然后累加计数，当达到计数最大值后，然后从 0 再次开始计数。

在 PreLoad 模式下，计数器的初始值是 PreLoad 寄存器的值，然后向上累加计数，当满足 PreLoad 条件时，计数器的值被置为 PreLoad 寄存器的值，然后计数器再次开始向上累加计数，在定时器的计数器计数过程中，一旦计数器的值与三个比较器中的某比较值一致，该比较器的比较标志就会置位，并可以产生相应的比较中断。

配置寄存器 TCMR 中 timer2\_mode 设置 TMR2 的计数模式，配置寄存器 TCMR 中 timer3\_mode 设置 TMR3 的计数模式。

若预加载寄存器的值为 10，比较器 0 的值为 13，比较器 1 的值为 16，比较器 2 的值为 19，则定时器在 PreLoad 的模式下工作时序如下图：

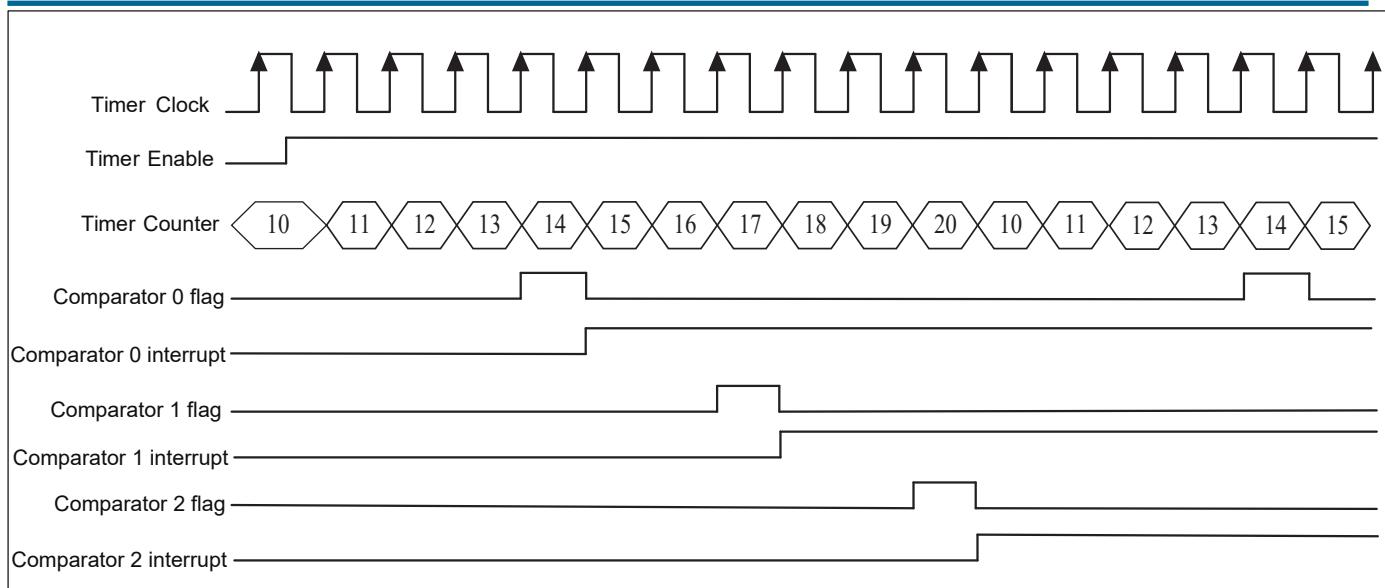


图 19.3: 定时器在 PreLoad 模式下工作时序

在 FreeRun 模式下，定时器工作时序与 PreLoad 基本相同，只是计数器会从 0 开始累计到最大值，期间产生的比较标志和比较中断的机制与 FreeRun 模式相同。

TMR2 可以使用内部的时钟源计算外部 gpio 的脉冲宽度。通过设置寄存器 GPIO 中 timer2\_gpio\_en 开启该功能，通过设置 timer2\_gpio\_inv 位，判断获取的是外部 gpio 的高电平还是低电平的宽度，如果该位为 0，表示高电平；如果该位为 1，表示低电平；另外需要将外部的 gpio 功能设置为 gpio\_tmr\_clk 功能。通过配置 GLB 模块中寄存器 dig\_clk\_cfg2 中的 gpio\_tmr\_clk\_sel[13:12] 位；同时需要将寄存器 dig\_clk\_cfg2[11:8] 中的某一位配置为 0，需要和 gpio\_tmr\_clk\_sel 配套使用。具体如下：

- 若 gpio\_tmr\_clk\_sel[13:12] 配置为 0，则寄存器 dig\_clk\_cfg2 中的 chip\_clk\_out\_0\_en 设置为 0
- 若 gpio\_tmr\_clk\_sel[13:12] 配置为 1，则寄存器 dig\_clk\_cfg2 中的 chip\_clk\_out\_1\_en 设置为 0
- 若 gpio\_tmr\_clk\_sel[13:12] 配置为 2，则寄存器 dig\_clk\_cfg2 中的 chip\_clk\_out\_2\_en 设置为 0
- 若 gpio\_tmr\_clk\_sel[13:12] 配置为 3，则寄存器 dig\_clk\_cfg2 中的 chip\_clk\_out\_3\_en 设置为 0

配置完成后，使能 timer。当寄存器 GPIO 中 gpio\_lat\_ok 置 1 后，获取寄存器 GPIO\_LAT2 和寄存器 GPIO\_LAT1 的值。外部 gpio 的脉冲宽度的计算方式：(GPIO\_LAT2-GPIO\_LAT1) \* timer 内部时钟源的 1 个周期的宽度；

例如：timer 的内部时钟源为 80M，外部 gpio 的频率为 2M，且占空比为 1: 1，将 timer2\_gpio\_inv 位写 1，表示计算外部 gpio 低电平的宽度。按照上述配置完成后，得到寄存器 GPIO\_LAT2 和寄存器 GPIO\_LAT1 的差为 20，则外部 gpio 的低电平宽度为： $20 * (1 / 80000000) = 1 / 4000000$ ；将 timer2\_gpio\_inv 位写 0，表示计算外部 gpio 高电平的宽度。按照上述配置完成后，得到寄存器 GPIO\_LAT2 和寄存器 GPIO\_LAT1 的差为 20，则外部 gpio 的高电平宽度为： $20 * (1 / 80000000) = 1 / 4000000$ ；

### 19.3.2 看门狗定时器工作原理

Watchdog 定时器包含一个计数器和一个比较器，计数器从 0 开始累加计数，如果计数器被复位（喂狗），则从 0 再次开始向上计数，当计数器的值与比较器相等的时候，可以产生一个比较中断信号或者系统复位信号，用户可以根据需要选择使用其中一个。看门狗计数器会在每个计数周期单位上加 1，软件可以在任何时间点通过 APB 将看门狗计数器归零。

配置寄存器 WMR 中 wmr 设置比较值，

若比较器的值为 6, Watchdog 的工作时序如下图所示

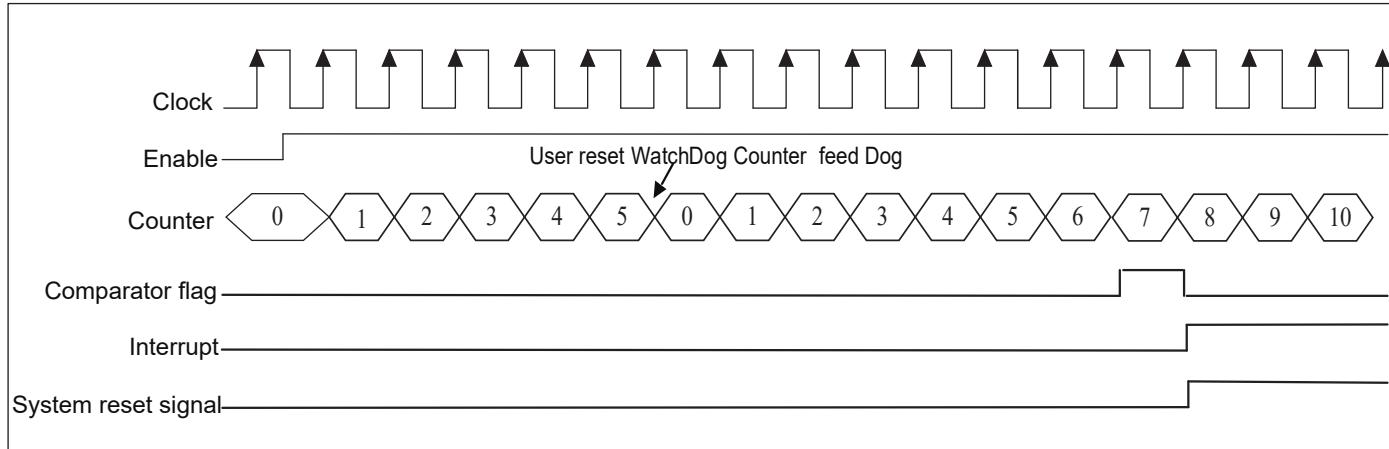


图 19.4: Watchdog 工作时序

### 19.3.3 报警设定

每一组计数器有三个比较值提供软件设定，并可设定每一组比较值是否触发报警中断，当计数器与比较值吻合且设定会报警时，计数器会通过中断通知处理器。软件可以通过 APB 读取目前是否发生报警和是哪个比较值触发报警中断，当清理报警中断时亦会同步清理报警状态。

### 19.3.4 看门狗报警

每个计数器可设定一组比较值，当软件因为系统错误，来不及将看门狗计数器归零，导致看门狗计数器超过比较值时，便会触发看门狗报警，报警方式有两种，第一种是通过中断通知软件进行必要的处置，第二种是进入系统看门狗复位，看门狗复位被触发时，会通知系统复位控制器，并做好系统复位前准备，当一切就绪后进入系统勘门狗复位，值得注意的是，软件可通过 APB 读取 WSR 寄存器得知是否曾经发生看门狗系统复位。

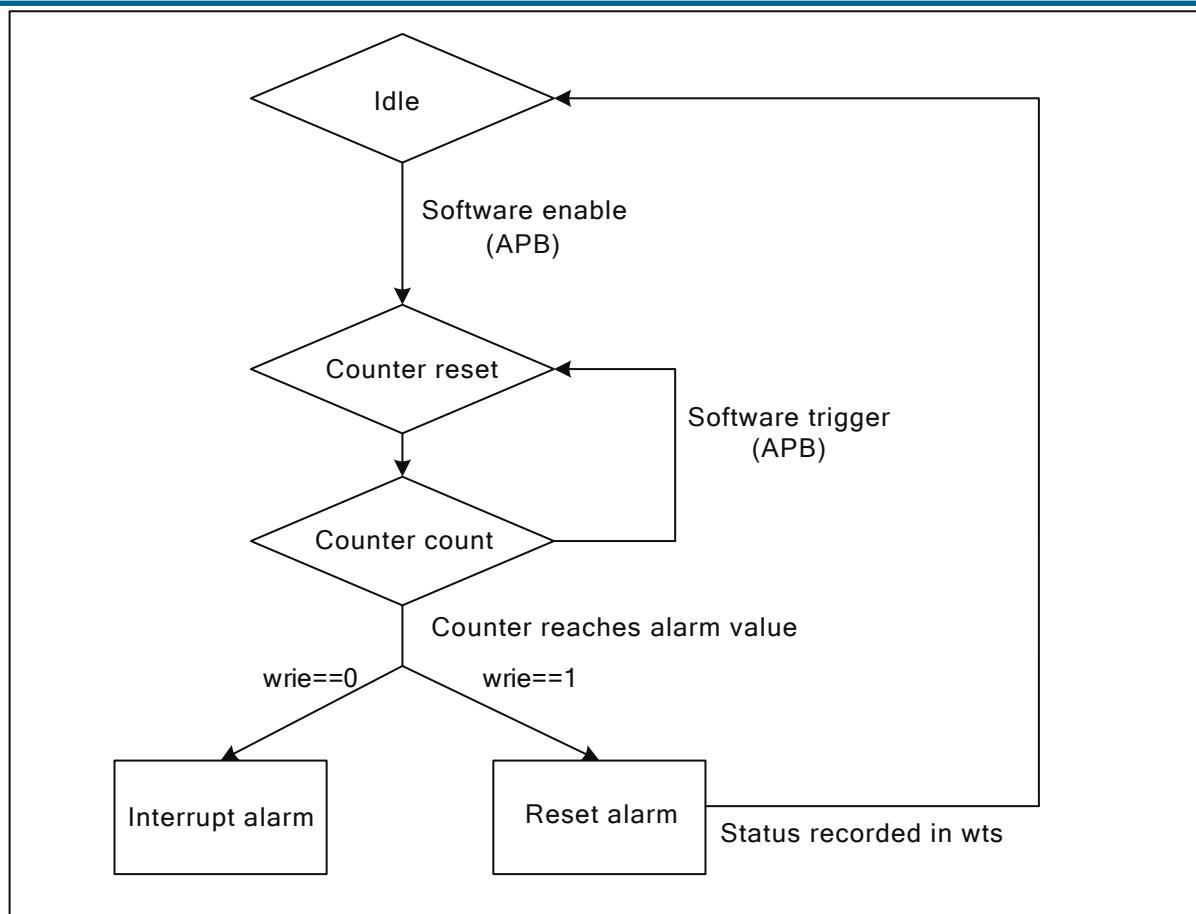


图 19.5: 看门狗报警机制

## 19.4 寄存器描述

名称	描述
TCCR	Timer Clock Source
TMR2_0	Timer2 Match Value 0
TMR2_1	Timer2 Match Value 1
TMR2_2	Timer2 Match Value 2
TMR3_0	Timer3 Match Value 0
TMR3_1	Timer3 Match Value 1
TMR3_2	Timer3 Match Value 2
TCR2	Timer2 Counter Value
TCR3	Timer3 Counter Value
TSR2	Timer2 Match Status
TSR3	Timer3 Match Status

名称	描述
TIER2	Timer2 Match Interrupt Enable
TIER3	Timer3 Match Interrupt Enable
TPLVR2	Timer2 Pre-Load Value
TPLVR3	Timer3 Pre-Load Value
TPLCR2	Timer2 Pre-Load Control
TPLCR3	Timer3 Pre-Load Control
WMER	Watch-dog reset/interrupt Mode
WMR	Watch-dog Match Value
WVR	Watch-dog Counter Value
WSR	Watch-dog Reset Status
TICR2	Timer2 Interrupt Clear
TICR3	Timer3 Interrupt Clear
WICR	WDT Interrupt Clear
TCER	Timer Counter Enable/Clear
TCMR	Timer Counter Mode
TILR2	Timer2 Match Interrupt Mode
TILR3	Timer3 Match Interrupt Mode
WCR	WDT Counter Reset
WFAR	WDT Access Key1
WSAR	WDT Access Key2
TCVWR2	Timer2 Counter Latch Value
TCVWR3	Timer3 Counter Latch Value
TCVSYN2	Timer2 Counter Sync Value
TCVSYN3	Timer3 Counter Sync Value
TCDR	Timer Division
GPIO	GPIO Mode
GPIO_LAT1	GPIO Latch Value1
GPIO_LAT2	GPIO Latch Value2
TCDR_FORCE	Timer Division Force

### 19.4.1 TCCR

地址: 0x30009000

ID								tmr_rsv							
31 30 29 28 27 26 25 24								23 22 21 20 19 18 17 16							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD CS\_wdt CS\_3 CS\_2

位	名称	权限	复位值	描述
31:24	ID	r	8'ha5	
23:16	tmr_rsv	rsvd	0	
15:12	RSVD			
11:8	cs_wdt	r/w	4'd1	WDT 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock
7:4	cs_3	r/w	4'd5	Timer3 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock
3:0	cs_2	r/w	4'd5	Timer2 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock

### 19.4.2 TMR2\_0

地址: 0x30009010

tmr2_0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr2\_0

位	名称	权限	复位值	描述
31:0	tmr2_0	r/w	32'hffffffff	Timer2 Match Value 0

### 19.4.3 TMR2\_1

地址: 0x30009014

tmr2\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr2\_1

位	名称	权限	复位值	描述
31:0	tmr2_1	r/w	32'hffffffff	Timer2 Match Value 1

### 19.4.4 TMR2\_2

地址: 0x30009018

tmr2\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr2\_2

位	名称	权限	复位值	描述
31:0	tmr2_2	r/w	32'hffffffff	Timer2 Match Value 2

### 19.4.5 TMR3\_0

地址: 0x3000901c

tmr3\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr3\_0

位	名称	权限	复位值	描述
31:0	tmr3_0	r/w	32'hffffffff	Timer3 Match Value 0

### 19.4.6 TMR3\_1

地址: 0x30009020

tmr3\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr3\_1

位	名称	权限	复位值	描述
31:0	tmr3_1	r/w	32'hffffffff	Timer3 Match Value 1

### 19.4.7 TMR3\_2

地址: 0x30009024

tmr3\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr3\_2

位	名称	权限	复位值	描述
31:0	tmr3_2	r/w	32'hffffffff	Timer3 Match Value 2

### 19.4.8 TCR2

地址: 0x3000902c

tcr2\_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tcr2\_cnt

位	名称	权限	复位值	描述
31:0	tcr2_cnt	r	0	Timer2 Counter Value

### 19.4.9 TCR3

地址: 0x30009030

tcr3\_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tcr3\_cnt

位	名称	权限	复位值	描述
31:0	tcr3_cnt	r	0	Timer3 Counter Value

### 19.4.10 TSR2

地址: 0x30009038

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

位	名称	权限	复位值	描述
31:3	RSVD			
2	tsr2_2	r	0	Timer2 match value 2 status/Clear interrupt would also clear this bit
1	tsr2_1	r	0	Timer2 match value 1 status/Clear interrupt would also clear this bit
0	tsr2_0	r	0	Timer2 match value 0 status/Clear interrupt would also clear this bit

### 19.4.11 TSR3

地址: 0x3000903c

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD tsr3\_2 tsr3\_1 tsr3\_0

位	名称	权限	复位值	描述
31:3	RSVD			
2	tsr3_2	r	0	Timer3 match value 2 status/Clear interrupt would also clear this bit
1	tsr3_1	r	0	Timer3 match value 1 status/Clear interrupt would also clear this bit
0	tsr3_0	r	0	Timer3 match value 0 status/Clear interrupt would also clear this bit

#### 19.4.12 TIER2

地址: 0x300009044

RSVD	tier2_2	tier2_1	tier2_0													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

位	名称	权限	复位值	描述
31:3	RSVD			
2	tier2_2	r/w	0	Timer2 match value 2 interrupt enable
1	tier2_1	r/w	0	Timer2 match value 1 interrupt enable
0	tier2_0	r/w	0	Timer2 match value 0 interrupt enable

### 19.4.13 TIER3

地址: 0x30009048

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |      |      |

RSVD tier3\_2 tier3\_1 tier3\_0

位	名称	权限	复位值	描述
31:3	RSVD			
2	tier3_2	r/w	0	Timer3 match value 2 interrupt enable
1	tier3_1	r/w	0	Timer3 match value 1 interrupt enable
0	tier3_0	r/w	0	Timer3 match value 0 interrupt enable

### 19.4.14 TPLVR2

地址: 0x30009050

tplvr2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

tplvr2

位	名称	权限	复位值	描述
31:0	tplvr2	r/w	0	Timer2 Pre-Load Value

### 19.4.15 TPLVR3

地址: 0x30009054

tplvr3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

tplvr3

位	名称	权限	复位值	描述
31:0	tplvr3	r/w	0	Timer3 Pre-Load Value

## 19.4.16 TPLCR2

地址: 0x3000905c

位	名称	权限	复位值	描述
31:2	RSVD			
1:0	tplcr2	r/w	0	Timer2 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

19.4.17 TPLCR3

地址: 0x30009060

位	名称	权限	复位值	描述
31:2	RSVD			

位	名称	权限	复位值	描述
1:0	tplcr3	r/w	0	Timer3 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

#### 19.4.18 WMER

地址: 0x30009064

RSVD	wrie	we														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

位	名称	权限	复位值	描述
31:2	RSVD			
1	wrie	r/w	0	WDT reset/interrupt mode 1'b0 - WDT expiration to generate interrupt 1'b1 - WDT expiration to generate reset source
0	we	r/w	0	WDT enable register

#### 19.4.19 WMR

地址: 0x30009068

RSVD	wdt_align															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

wmr

位	名称	权限	复位值	描述
31:17	RSVD			

位	名称	权限	复位值	描述
16	wdt_align	r/w	0	WDT compare value update align interrupt
15:0	wmr	r/w	16'hffff	WDT counter match value

#### 19.4.20 WVR

地址: 0x3000906c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

wdt\_cnt

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	wdt_cnt	r	0	WDT counter value

#### 19.4.21 WSR

地址: 0x30009070

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

wts

位	名称	权限	复位值	描述
31:1	RSVD			
0	wts	w	0	WDT reset status Write 0 to clear the WDT reset status Read 1 indicates reset was caused by the WDT

### 19.4.22 TICR2

地址: 0x30009078

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD tclr2\_2 tclr2\_1 tclr2\_0

位	名称	权限	复位值	描述
31:3	RSVD			
2	tclr2_2	w	0	Timer2 Interrupt clear for match comparator 2
1	tclr2_1	w	0	Timer2 Interrupt clear for match comparator 1
0	tclr2_0	w	0	Timer2 Interrupt clear for match comparator 0

### 19.4.23 TICR3

地址: 0x3000907c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD tclr3\_2 tclr3\_1 tclr3\_0

位	名称	权限	复位值	描述
31:3	RSVD			
2	tclr3_2	w	0	Timer3 Interrupt clear for match comparator 2
1	tclr3_1	w	0	Timer3 Interrupt clear for match comparator 1
0	tclr3_0	w	0	Timer3 Interrupt clear for match comparator 0

### 19.4.24 WICR

地址: 0x30009080

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	wiclr

位	名称	权限	复位值	描述
31:1	RSVD			
0	wiclr	w	0	WDT Interrupt Clear

### 19.4.25 TCER

地址: 0x30009084

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	tcr3_cnt_clr tcr2_cnt_clr timer3_en timer2_en RSVD

位	名称	权限	复位值	描述
31:7	RSVD			
6	tcr3_cnt_clr	r/w	0	Timer3 count clear
5	tcr2_cnt_clr	r/w	0	Timer2 count clear
4:3	RSVD			
2	timer3_en	r/w	0	Timer3 count enable
1	timer2_en	r/w	0	Timer2 count enable
0	RSVD			

### 19.4.26 TCMR

地址: 0x30009088

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD										
										timer3_align	timer2_align	RSVD	RSVD	timer3_mode	timer2_mode	RSVD

位	名称	权限	复位值	描述
31:7	RSVD			
6	timer3_align	r/w	0	Timer3 compare value update align interrupt
5	timer2_align	r/w	0	Timer2 compare value update align interrupt
4:3	RSVD			
2	timer3_mode	r/w	0	0:pre-load mode 1:free run mode
1	timer2_mode	r/w	0	0:pre-load mode 1:free run mode
0	RSVD			

### 19.4.27 TILR2

地址: 0x30009090

RSVD	RSVD	RSVD														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RSVD	tilr2_2	tilr2_1	tilr2_0													

位	名称	权限	复位值	描述
31:3	RSVD			
2	tilr2_2	r/w	0	0:level 1:edge
1	tilr2_1	r/w	0	0:level 1:edge
0	tilr2_0	r/w	0	0:level 1:edge

### 19.4.28 TILR3

地址: 0x30009094

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

位	名称	权限	复位值	描述
31:3	RSVD			
2	tilr3_2	r/w	0	0:level 1:edge
1	tilr3_1	r/w	0	0:level 1:edge
0	tilr3_0	r/w	0	0:level 1:edge

### 19.4.29 WCR

地址: 0x30009098

RSVD																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

位	名称	权限	复位值	描述
31:1	RSVD			
0	wcr	w	0	WDT Counter Reset

### 19.4.30 WFAR

地址: 0x3000909c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

wfar

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	wfar	w	0	WDT access key1 - 16'hBABA

#### 19.4.31 WSAR

地址: 0x300090a0

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

wsar

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	wsar	w	0	WDT access key2 - 16'hEB10

#### 19.4.32 TCVWR2

地址: 0x300090a8

tcr2_cnt_lat																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

tcr2\_cnt\_lat

位	名称	权限	复位值	描述
31:0	tcr2_cnt_lat	r	0	Timer2 Counter Latch Value

#### 19.4.33 TCVWR3

地址: 0x300090ac

tcr3_cnt_lat																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

tcr3\_cnt\_lat

位	名称	权限	复位值	描述
31:0	tcr3_cnt_lat	r	0	Timer3 Counter Latch Value

#### 19.4.34 TCVSYN2

地址: 0x300090b4

tcr2\_cnt\_sync

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tcr2\_cnt\_sync

位	名称	权限	复位值	描述
31:0	tcr2_cnt_sync	r	0	Timer2 Counter Sync Value (continue readable)

#### 19.4.35 TCVSYN3

地址: 0x300090b8

tcr3\_cnt\_sync

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tcr3\_cnt\_sync

位	名称	权限	复位值	描述
31:0	tcr3_cnt_sync	r	0	Timer3 Counter Sync Value (continue readable)

#### 19.4.36 TCDR

地址: 0x300090bc

wcdr

tcdr3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

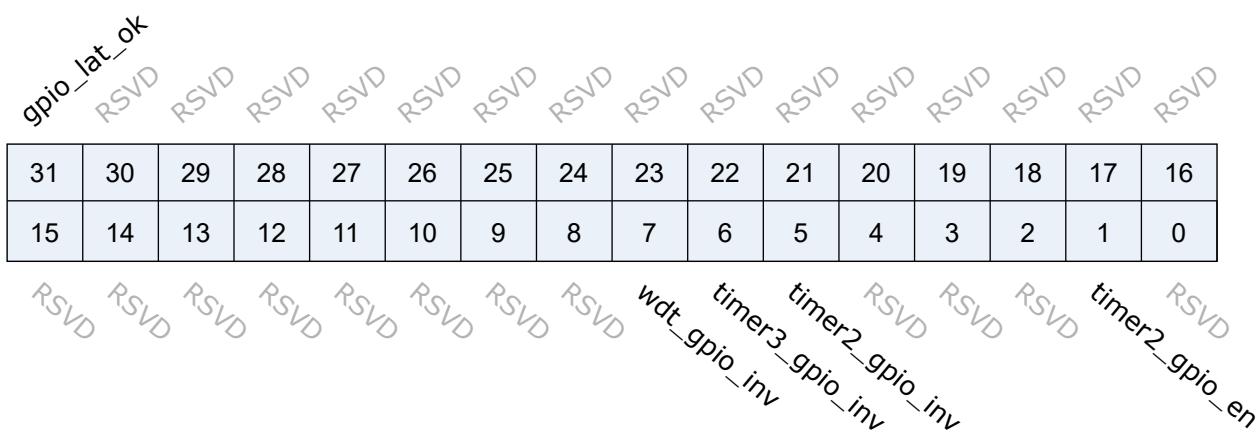
tcdr2

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

位	名称	权限	复位值	描述
31:24	wcdr	r/w	0	WDT clock division value register
23:16	tcdr3	r/w	0	Timer3 clock division value register
15:8	tcdr2	r/w	0	Timer2 clock division value register
7:0	RSVD			

#### 19.4.37 GPIO

地址: 0x300090c0



位	名称	权限	复位值	描述
31	gpio_lat_ok	r	0	Latch Done. Pulse width = (GPIO_LAT2 - GPIO_LAT1) * (Timer2 Cycle)
30:8	RSVD			
7	wdt_gpio_inv	r/w	0	WDT gpio polarity 0:pos 1:neg
6	timer3_gpio_inv	r/w	0	Timer3 gpio polarity 0:pos 1:neg
5	timer2_gpio_inv	r/w	0	Timer2 gpio polarity 0:pos 1:neg
4:2	RSVD			
1	timer2_gpio_en	r/w	0	Timer2 gpio measure enable
0	RSVD			

### 19.4.38 GPIO\_LAT1

地址: 0x300090c4

gpio\_lat1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpio\_lat1

位	名称	权限	复位值	描述
31:0	gpio_lat1	r	0	Pos-Edge Latch Timer2

### 19.4.39 GPIO\_LAT2

地址: 0x300090c8

gpio\_lat2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpio\_lat2

位	名称	权限	复位值	描述
31:0	gpio_lat2	r	0	Neg-Edge Latch Timer2

### 19.4.40 TCDR\_FORCE

地址: 0x300090cc

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

RSVD	wcdrl_force	tcdrl2_force	tcdrl3_force	RSVD												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

位	名称	权限	复位值	描述
31:5	RSVD			

位	名称	权限	复位值	描述
4	wcdr_force	r/w	0	Force WDT clock division value to counter
3	RSVD			
2	tcdr3_force	r/w	0	Force Timer3 clock division value to counter
1	tcdr2_force	r/w	0	Force Timer2 clock division value to counter
0	RSVD			

## 20.1 简介

I2S 全称 Inter-IC Sound, Integrated Interchip Sound, 或简写 IIS, 是飞利浦在 1986 年定义（1996 年修订）的数字音频传输标准，用于数字音频数据在系统内部器件之间传输。I2S 将时钟信号与数据信号分开传输，使得接收端不用从数据信号还原时钟，进而降低接收端的设计难度。

## 20.2 主要特征

- 支持主模式以及从模式
- 支持 Left-Justified/Right-Justified/Normal I2S/DSP 等数据格式
- 支持 8/16/24/32 比特数据宽度
- 支持数据 MSB/LSB 切换
- 支持 DMA 传输模式
- 除单声道/双声道模式之外，同时支持四声道与六声道模式
- 支持播放单声道音频复制为双声道模式
- 支持低于 16bits 双声道录音数据合并为低于 32bits FIFO 宽度数据
- 支持 16/32/48/64 bit frame size
- 支持动态静音切换功能
- 数据发送 FIFO 的宽度为 32 位，深度 16
- 数据接收 FIFO 的宽度为 32 位，深度 16

## 20.3 功能描述

引脚列表：

表 20.1: I2S 引脚

名称	类型	描述
I2Sx_DI	输入	串行数据输入
I2Sx_DO	输出	串行数据输出
I2Sx_BCLK	输入/输出	同步传输时钟，作为主机时为输出，作为从机时为输入
I2Sx_FS	输入/输出	数据起始/结束表示信号，作为主机时为输出，作为从机时为输入

## 20.4 功能描述

### 20.4.1 数据格式描述

I2S 模块支持标准 I2S 协议，左对齐模式，右对齐模式。标准 I2S 是左对齐模式的特殊情况。模式由 I2S\_CONFIG[17:16] 配置。

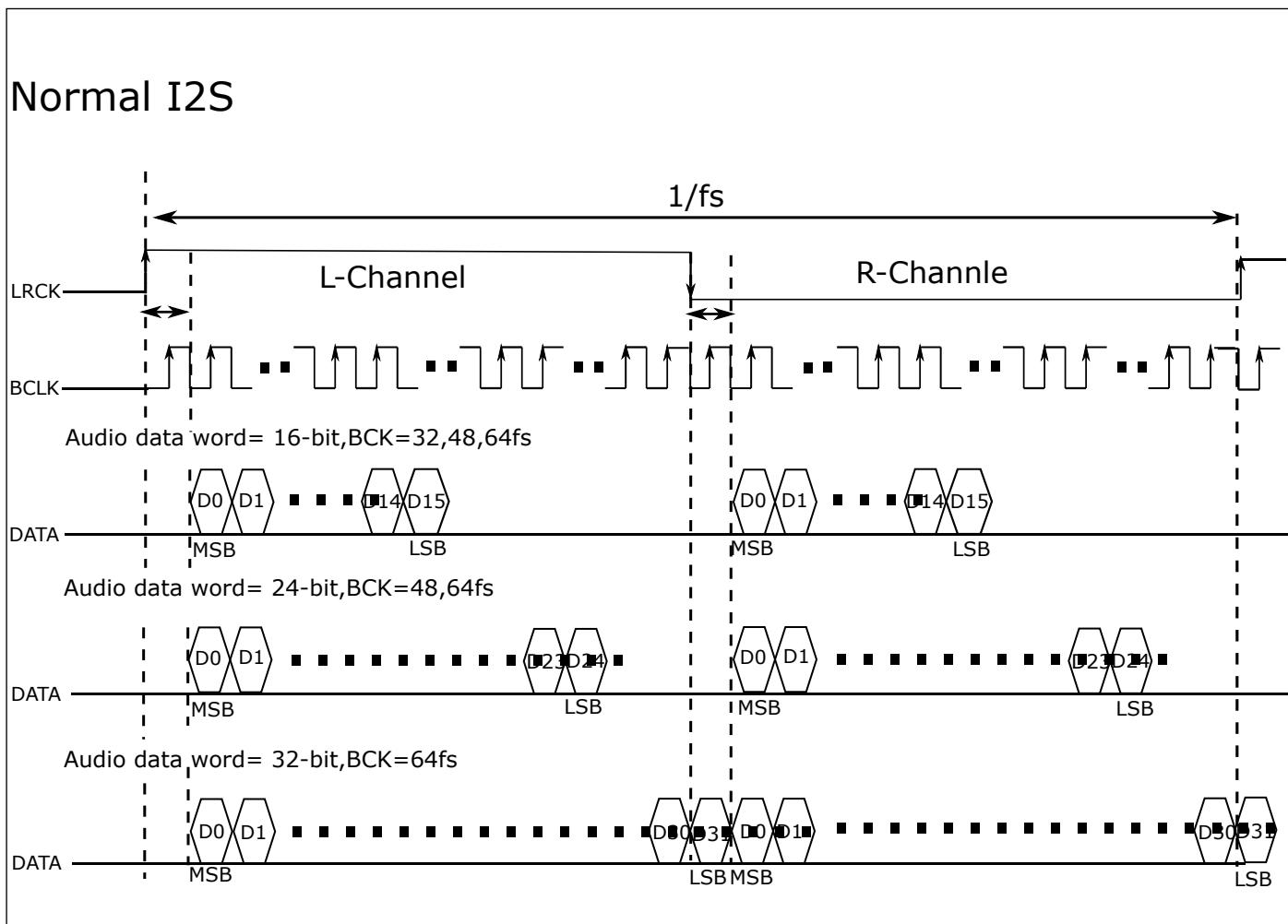


图 20.1: 标准 I2S 数据格式

由 I2S\_CONFIG[25:20] 来配置 OFFSET, 左对齐模式与标准 I2S 的唯一区别在于对 I2S\_CONFIG[25:20] 的配置不同。

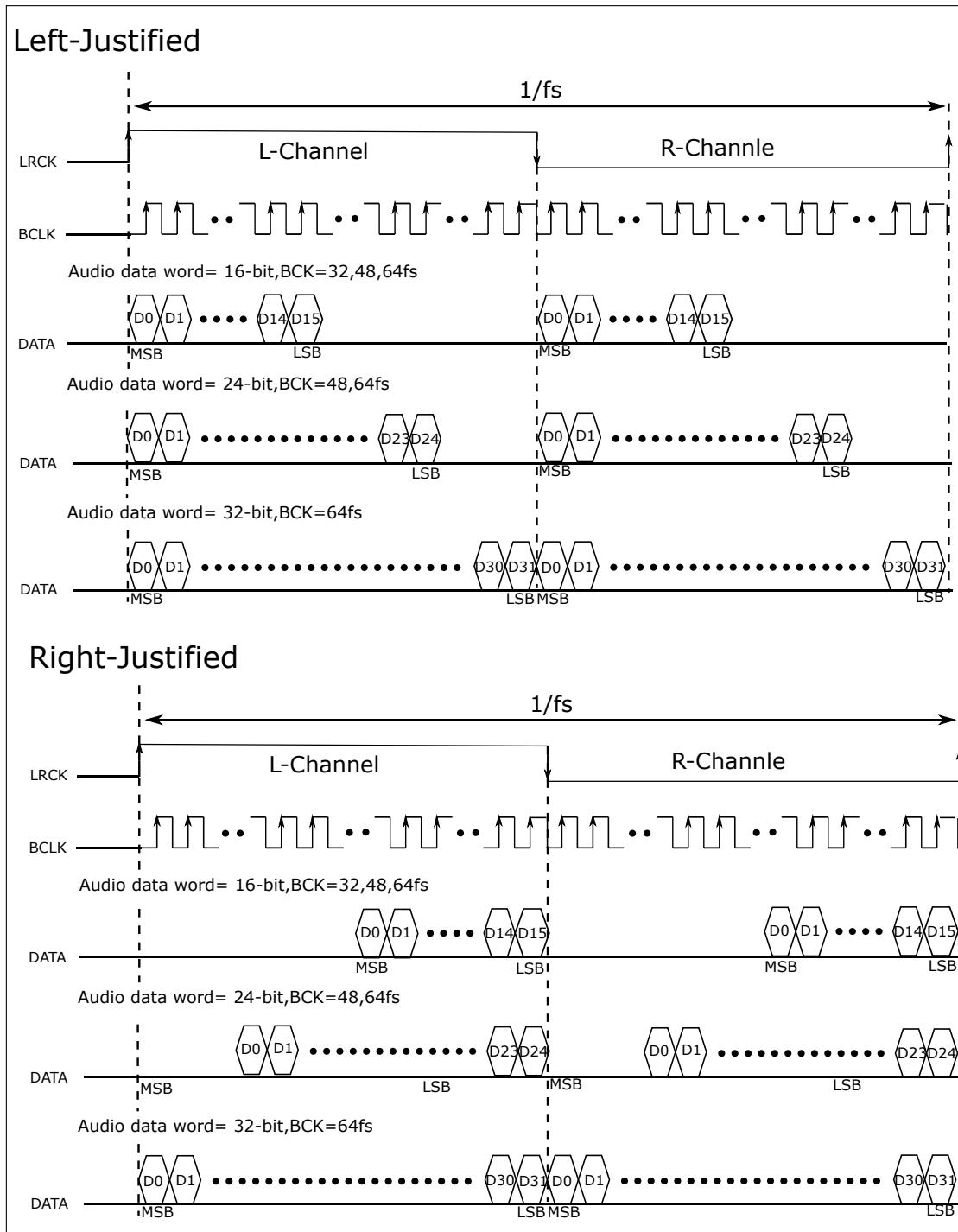


图 20.2: I2S 左对齐/右对齐数据格式

## DSP-MTK TDM64

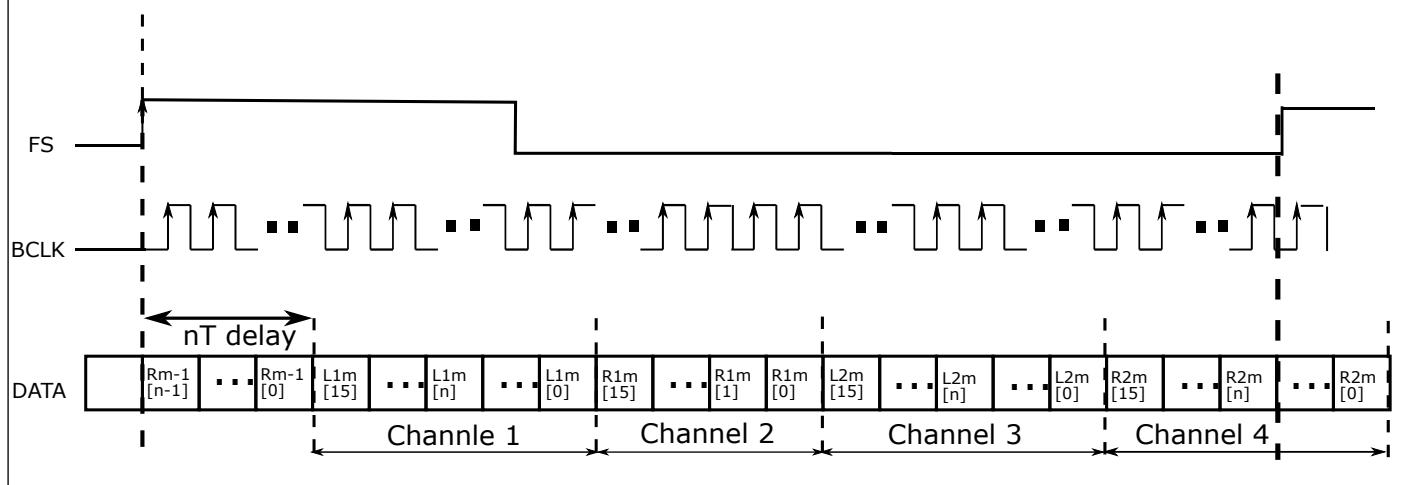


图 20.3: I2S TDM64 模式六通道录音

可以通过 I2S\_CONFIG[6] 来控制单个脉冲的宽度，当选择为 0 时，FS 信号线高电平脉冲的宽度为 data size 的宽度，当选择为 1 的时候，FS 信号线高电平脉冲的宽度为 1。一般情况下，多通道的 TDM64 模式，此寄存器配置为 1。

### 20.4.2 基本架构图

### 20.4.3 时钟源

I2S 的时钟源由 audio PLL 提供，时钟中的分频器用于对时钟源进行分频，然后产生时钟信号来驱动 I2S 模块。如下图所示：

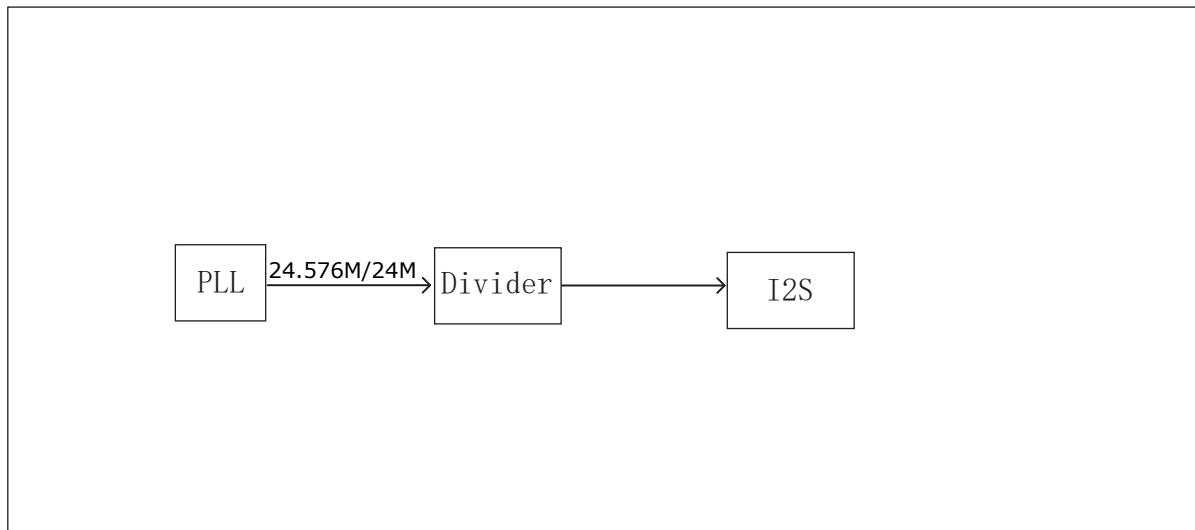


图 20.4: I2S 时钟

## 20.4.4 I2S 中断

I2S 有着丰富的中断控制，包括以下几种中断模式：

- TX FIFO 请求中断
- RX FIFO 请求中断
- overrun/underrun error 中断

当 I2S\_FIFO\_CONFIG\_1 中 TX\_FIFO\_CNT 大于 TX\_FIFO\_TH 时，产生 TX FIFO 请求中断。当条件不满足时该中断标志会自动清除。

当 I2S\_FIFO\_CONFIG\_1 中 RX\_FIFO\_CNT 大于 RX\_FIFO\_TH 时，产生 RX FIFO 请求中断。当条件不满足时该中断标志会自动清除。

如果 TX/RX FIFO 发生了上溢或者下溢，会触发 error 中断，当异常消失后，标志位会自动清空。

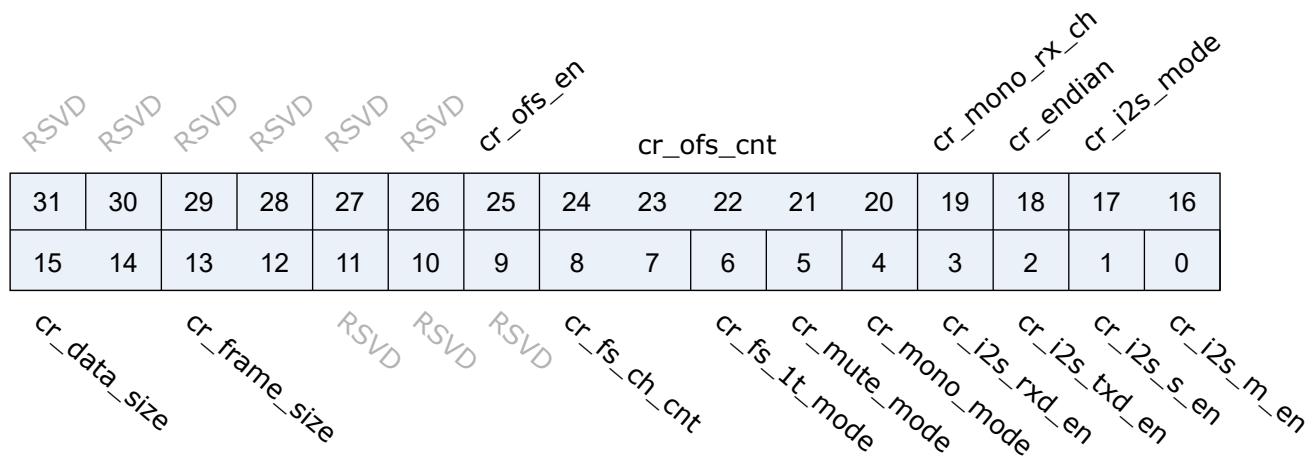
I2S 的所有中断的使能位以及中断标志位都在 I2S\_INT\_STS 寄存器。

## 20.5 寄存器描述

名称	描述
i2s_config	
i2s_int_sts	
i2s_bclk_config	
i2s_fifo_config_0	
i2s_fifo_config_1	
i2s_fifo_wdata	
i2s_fifo_rdata	
i2s_io_config	

## 20.5.1 i2s\_config

地址: 0x2000ab00



位	名称	权限	复位值	描述
31:26	RSVD			
25	cr_ofs_en	r/w	1'b0	Offset enable 1'b0: Disabled, 1'b1: Enabled
24:20	cr_ofs_cnt	r/w	5'd0	Offset cycle count (unit: cycle of I2S BCLK) 5'd0: 1 cycle 5'd1: 2 cycles ...
19	cr_mono_rx_ch	r/w	1'b0	RX mono mode channel select signal 1'b0: L-channel 1'b1: R-channel
18	cr_endian	r/w	1'b0	Data endian (bit reverse) 1'b0: MSB goes out first, 1'b1: LSB goes out first
17:16	cr_i2s_mode	r/w	2'd0	2'd0: Left-Justified, 2'd1: Right-Justified, 2'd2: DSP, 2'd3: Reserved
15:14	cr_data_size	r/w	2'd1	Data bit width of each channel 2'd0: 8, 2'd1: 16, 2'd2: 24, 2'd3: 32 (bits)
13:12	cr_frame_size	r/w	2'd1	Frame size of each channel 2'd0: 8, 2'd1: 16, 2'd2: 24, 2'd3: 32 (cycles)
11:9	RSVD			

位	名称	权限	复位值	描述
8:7	cr_fs_ch_cnt	r/w	2'd0	Channel count of each frame 2'd0: FS 2-channel mode 2'd1: FS 3-channel mode (DSP mode only) 2'd2: FS 4-channel mode (DSP mode only) 2'd3: FS 6-channel mode (DSP mode only) Note: cr_mono_mode & cr_fifo_lr_merge will be invalid in 3-channel mode Note: frame_size must equal data_size in 3/4/6-channel mode
6	cr_fs_1t_mode	r/w	1'b0	1'b0: FS high/low is even, 1'b1: FS only asserts for 1 cycle
5	cr_mute_mode	r/w	1'b0	1'b0: Normal mode, 1'b1: Mute mode
4	cr_mono_mode	r/w	1'b0	1'b0: Stereo mode, 1'b1: Mono mode Note: csr_mono_mode & csr_fifo_lr_merge should NOT be enabled at the same time
3	cr_i2s_rxd_en	r/w	1'b0	Enable signal of I2S RXD signal
2	cr_i2s_txd_en	r/w	1'b0	Enable signal of I2S TXD signal
1	cr_i2s_s_en	r/w	1'b0	Enable signal of I2S Slave function, cannot enable both csr_i2s_m_en & csr_i2s_s_en
0	cr_i2s_m_en	r/w	1'b0	Enable signal of I2S Master function, cannot enable both csr_i2s_m_en & csr_i2s_s_en

## 20.5.2 i2s\_int\_sts

地址: 0x2000ab04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD      RSVD      RSVD      RSVD      RSVD      cr\_i2s\_fer\_en      cr\_i2s\_rxf\_en      cr\_i2s\_txf\_en  
 RSVD      RSVD

RSVD      RSVD      RSVD      RSVD      RSVD      cr\_i2s\_fer\_mask      cr\_i2s\_rxf\_mask      cr\_i2s\_txf\_mask  
 RSVD      RSVD

i2s\_fer\_int      i2s\_rxf\_int      i2s\_txf\_int

位	名称	权限	复位值	描述
31:27	RSVD			
26	cr_i2s_fer_en	r/w	1'b1	Interrupt enable of i2s_fer_int
25	cr_i2s_rxf_en	r/w	1'b1	Interrupt enable of i2s_rxf_int
24	cr_i2s_txf_en	r/w	1'b1	Interrupt enable of i2s_txf_int
23:11	RSVD			
10	cr_i2s_fer_mask	r/w	1'b1	Interrupt mask of i2s_fer_int
9	cr_i2s_rxf_mask	r/w	1'b1	Interrupt mask of i2s_rxf_int
8	cr_i2s_txf_mask	r/w	1'b1	Interrupt mask of i2s_txf_int
7:3	RSVD			
2	i2s_fer_int	r	1'b0	I2S TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	i2s_rxf_int	r	1'b0	I2S RX FIFO ready ( $rx\_fifo\_cnt > rx\_fifo\_th$ ) interrupt, auto-cleared when data is popped
0	i2s_txf_int	r	1'b1	I2S TX FIFO ready ( $tx\_fifo\_cnt > tx\_fifo\_th$ ) interrupt, auto-cleared when data is pushed

### 20.5.3 i2s\_bclk\_config

地址: 0x2000ab10

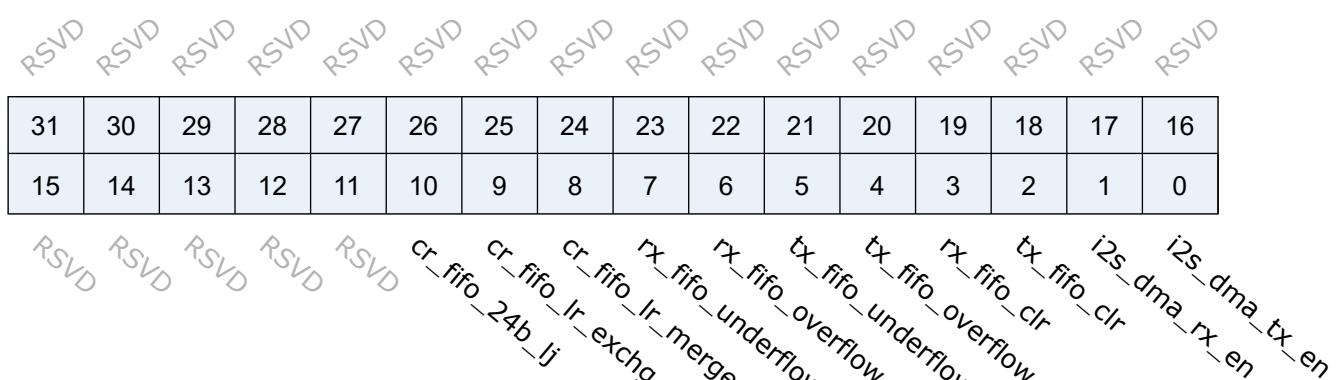
cr_bclk_div_h															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_bclk_div_l														
31:28	RSVD	RSVD	RSVD	RSVD										

位	名称	权限	复位值	描述
31:28	RSVD			
27:16	cr_bclk_div_h	r/w	12'd1	I2S BCLK active high period (unit: cycle of i2s_clk)
15:12	RSVD			
11:0	cr_bclk_div_l	r/w	12'd1	I2S BCLK active low period (unit: cycle of i2s_clk)

## 20.5.4 i2s\_fifo\_config\_0

地址: 0x2000ab80

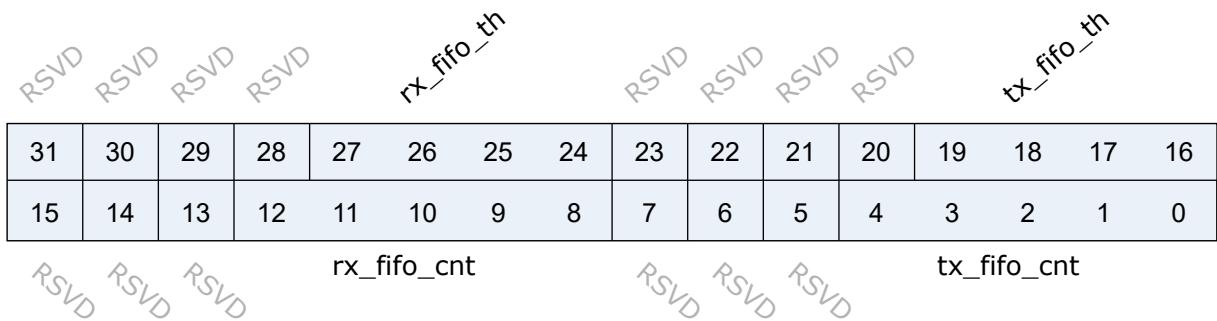


位	名称	权限	复位值	描述
31:11	RSVD			
10	cr_fifo_24b_lj	r/w	1'b0	FIFO 24-bit data left-justified mode 1'b0: Right-justified, 8'h0, data[23:0] 1'b1: Left-justified, data[23:0], 8'h0 Note: Valid only when cr_data_size = 2'd2 (24-bit)
9	cr_fifo_lr_exchg	r/w	1'b0	The position of L/R channel data within each entry is exchanged if this bit is enabled Can only be enabled if data size is 8 or 16 bits and csr_fifo_lr_merge is enabled
8	cr_fifo_lr_merge	r/w	1'b0	Each FIFO entry contains both L/R channel data if this bit is enabled Can only be enabled if data size is 8 or 16 bits Note: cr_fifo_lr_merge &cr_mono_mode should NOT be enabled at the same time Note: cr_fifo_lr_merge &cr_fifo_l_shift should NOT be enabled at the same time
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr

位	名称	权限	复位值	描述
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	i2s_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	i2s_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

### 20.5.5 i2s\_fifo\_config\_1

地址: 0x2000ab84



位	名称	权限	复位值	描述
31:28	RSVD			
27:24	rx_fifo_th	r/w	4'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:20	RSVD			
19:16	tx_fifo_th	r/w	4'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:13	RSVD			
12:8	rx_fifo_cnt	r	5'd0	RX FIFO available count
7:5	RSVD			
4:0	tx_fifo_cnt	r	5'd16	TX FIFO available count

### 20.5.6 i2s\_fifo\_wdata

地址: 0x2000ab88

i2s_fifo_wdata															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2s\_fifo\_wdata

位	名称	权限	复位值	描述
31:0	i2s_fifo_wdata	w	x	

### 20.5.7 i2s\_fifo\_rdata

地址: 0x2000ab8c

i2s\_fifo\_rdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2s\_fifo\_rdata

位	名称	权限	复位值	描述
31:0	i2s_fifo_rdata	r	32'h0	

### 20.5.8 i2s\_io\_config

地址: 0x2000abfc

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD

位	名称	权限	复位值	描述
31:8	RSVD			
7	cr_deg_en	r/w	1'b0	Deglitch enable (for all th input pins) 1'b0: Disabled, 1'b1: Enabled
6:4	cr_deg_cnt	r/w	3'd0	Deglitch cycle count (unit: cycle of I2S kernel clock) 3'd0: 1 cycle 3'd1: 2 cycles ...
3	cr_i2s_bclk_inv	r/w	1'b0	Inverse BCLK signal 0: No inverse, 1: Inverse

位	名称	权限	复位值	描述
2	cr_i2s_fs_inv	r/w	1'b0	Inverse FS signal 0: No inverse, 1: Inverse
1	cr_i2s_rxd_inv	r/w	1'b0	Inverse RXD signal 0: No inverse, 1: Inverse
0	cr_i2s_txd_inv	r/w	1'b0	Inverse TXD signal 0: No inverse, 1: Inverse

## 21.1 简介

芯片内置一个 PDM 音频处理模块，支持 pdm 接口麦克风录音

## 21.2 主要特征

- 集成 3 路 20bit ADC, 可支持 3 路数字 PDM 输入
  - 采样率: 8k~96k
  - 信噪比 (A-W): 97dB @ 0dB 增益
  - 谐波失真 + 噪声: -87dB @ 0dB 增益
  - 模拟前置增益: 0dB, 6~42 dB, 3dB 一档
- 可调节的高通滤波器和独立的数字音量控制，用于 ADC 通路
- 支持数字 PDM 接口，复用输入 GPIO
- 独立的数字音量控制
- 32 位宽度的发送、接受 FIFO
- 支持 DMA 传输模式

## 21.3 功能描述

PDM 模块基本框图如图所示。

```
.... figure:: ../../picture/PDMBasicStruct.pdf .. :align: center
```

PDM 模块包含了三路 PDM 数字接口解调器，经过抽取滤波器，HPF 滤波器之后，进入音量控制模块。用户可以通过音量控制模块，控制静音/非静音，控制音量大小，控制声音渐入/渐出效果。录音的数据最终会存放在深度为 32 的 FIFO 中，可以由 RX\_FIFO\_CTRL[25:24] 寄存器来控制 FIFO 的存放格式。PDM\_RX\_FIFO\_CTRL[5] 配置采样的分辨率。PDM 模块仅支持 PDM 音频数字接口输入。

### 21.3.1 PDM 中断

PDM 有着丰富的中断控制，包括以下几种中断模式：

- RX FIFO 请求中断
- RX FIFO underrun 中断
- RX FIFO overrun 中断

当 RX\_FIFO\_CTRL 中 RX\_DRQ\_CNT 大于 RX\_TRG\_LEVEL 时，产生 RX FIFO 请求中断。当条件不满足时该中断标志会自动清除。

当 RX FIFO 中并没有数据，但是用户却通过 RX\_FIFO\_CTRL 中的 RX\_CH\_EN 使能了 RX FIFO 调制，则会进入 RX FIFO underrun 中断。

当用户填入超过 RX FIFO 最大深度的数据的时候，会导致 RX FIFO 溢出，从而产生 RX FIFO overrun 中断。

### 21.3.2 FIFO 格式控制

PDM\_RX\_FIFO\_CTRL 可以控制音频数据储存在 FIFO 的格式。

用户通过配置 PDM\_RX\_FIFO\_CTRL[5] 来选择音频的分辨率。

当分辨率选择位 16bits 时，FIFO 控制器支持如下四种数据存储格式，由 FIFO\_CTRL[25:24] 来决定。

- Mode 0:  
 $\text{DATA}[31:0] = \{\text{FIFO}[19:14], 16'h0\}$
- Mode 1:  
 $\text{DATA}[31:0] = \{8\{\text{FIFO}[19]\}, \text{FIFO}[19:4], 8'h0\}$
- Mode 2:  
 $\text{DATA}[31:0] = \{12\{\text{FIFO}[19]\}, \text{FIFO}[19:4], 4'h0\}$
- Mode 3:  
 $\text{DATA}[31:0] = \{16\{\text{FIFO}[19]\}, \text{FIFO}[19:4]\}$

在录音/播放的时候，32Bits 数据的转换结果或待解调的数字音源在左边，记作 DATA[31:0]。他将被储存在 FIFO 中的格式如右边，因为录音的分辨率实际为 20bits，所以当选择分辨率为 16bits 的时候，需要对 20bit 的分辨率做一些剪裁，因此选取了 20bit 分辨率的高 16bits 作为最终的结果 FIFO[19:14]，并把它保存在了高 16bits 的位置，低 16bits 采用用 0 补齐的操作。Mode1、2、3 与 Mode0 的表示方式相同，值得说明的是，8{FIFO[19]} 符号表示的是会用 bit[19] 的值来填充高八位。

因此，当选择分辨率为 16bits 的时候，PDM 提供了四种模式来存放转换/输出的数字结果。

当分辨率为 20bits 的时候，四种模式的储存方式如下

- Mode 0:  
 $\text{DATA}[31:0] = \{\text{FIFO}[19:0], 12'h0\}$
- Mode 1:  
 $\text{DATA}[31:0] = \{8\{\text{FIFO}[19]\}, \text{FIFO}[19:0], 4'h0\}$
- Mode 2:  
 $\text{DATA}[31:0] = \{12\{\text{FIFO}[19]\}, \text{FIFO}[19:0]\}$

- Mode 3:

DATA[31:0] = {16{FIFO[19]}, FIFO[19:4]}

最高有效位的分布

- Mode 0:

有效数据的最高位在 31 bits

- Mode 1:

有效数据的最高位在 23 bits

- Mode 2:

有效数据的最高位在 19 bits

- Mode 3:

有效数据的最高位在 15 bits

### 21.3.3 FIFO 的启动与 DMA 搬运

PDM 的 FIFO 数据可以通过 DMA 进行搬运。

用户可以通过 PDM\_RX\_FIFO\_STATUS 寄存器实时获得目前 FIFO 有效数据的数量。

通过配置 FIFO\_CTRL[15:14] 来选择发起 DMA request 的 FIFO count 阈值，是 8/16/32，或者是由 FIFO\_CTRL[22:16] 配置来决定。

当 count 的值大于设定阈值，并且 PDM\_RX\_FIFO\_CTRL[12:8] 对应通路的 FIFO 被使能，则会发起一次 DMA 搬运。

注意，启动 TX FIFO 时，如果 TX FIFO 里面并没有有效的数据，则会触发 tx underrun 错误。因此要注意软件配置顺序。

## 21.4 寄存器描述

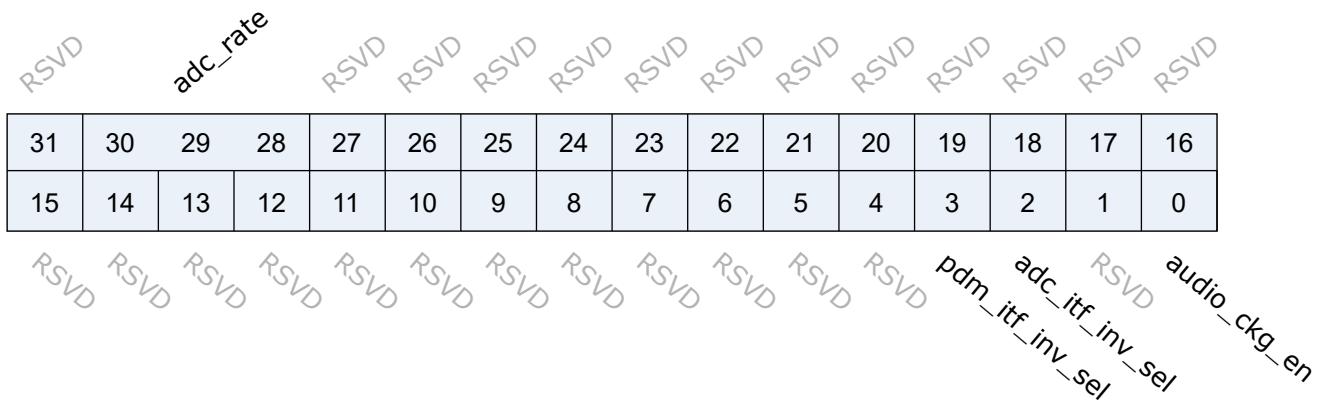
名称	描述
audpdm_top	
audpdm_if	
pdm_adc_0	
pdm_adc_1	
pdm_dac_0	
pdm_pdm_0	
pdm_rsvd0	
pdm_dbg_0	
pdm_dbg_1	
pdm_dbg_2	
pdm_dbg_3	



名称	描述
pdm_dbg_4	
pdm_adc_s0	
pdm_adc_s1	
pdm_adc_s2	
pdm_rx_fifo_ctrl	
pdm_rx_fifo_status	
pdm_rx_fifo_data	

## 21.4.1 audpdm\_top

地址: 0x2000ac00

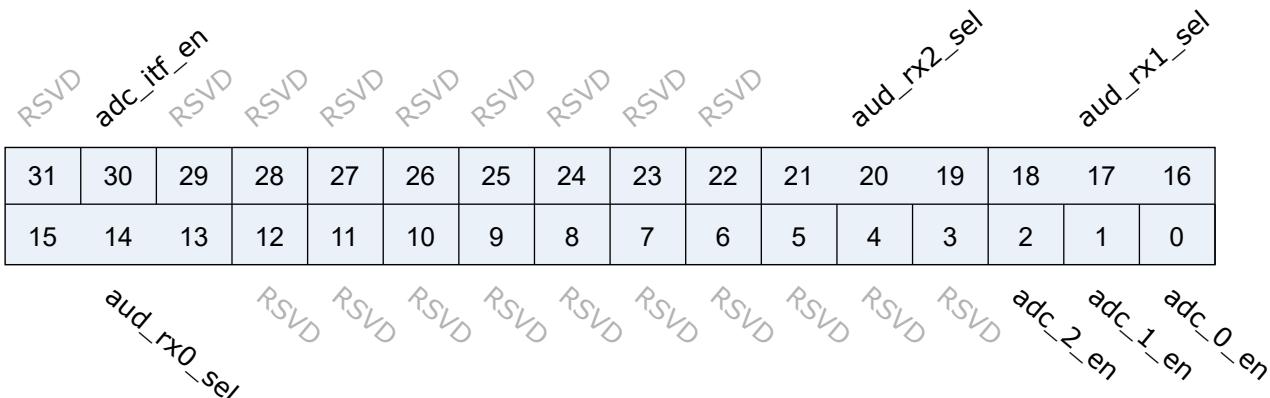


位	名称	权限	复位值	描述
31	RSVD			
30:28	adc_rate	r/w	3'd1	adc fs 0:8kHz, 1:16kHz, 2:24kHz(22.05k), 3:32kHz, 4:48kHz(44.1k), 5:96kHz, 6:reserved, 7:manual
27	RSVD			
26:24	dac_rate	r/w	3'd4	dac fs 0:8kHz, 1:16kHz, 2:24kHz(22.05k), 3:32kHz, 4:48kHz(44.1k), 5:96kHz, 6:192kHz, 7:manual
23:4	RSVD			
3	pdm_itf_inv_sel	r/w	1'd0	1:invert clk_pdm_inf
2	adc_itf_inv_sel	r/w	1'd0	1:invert clk_adc_itf

位	名称	权限	复位值	描述
1	dac_itf_inv_sel	r/w	1'd1	1:invert clk_dac_itf
0	audio_ckg_en	r/w	1'd0	1:enable audio clock generator

### 21.4.2 audpdm\_itf

地址: 0x2000ac04

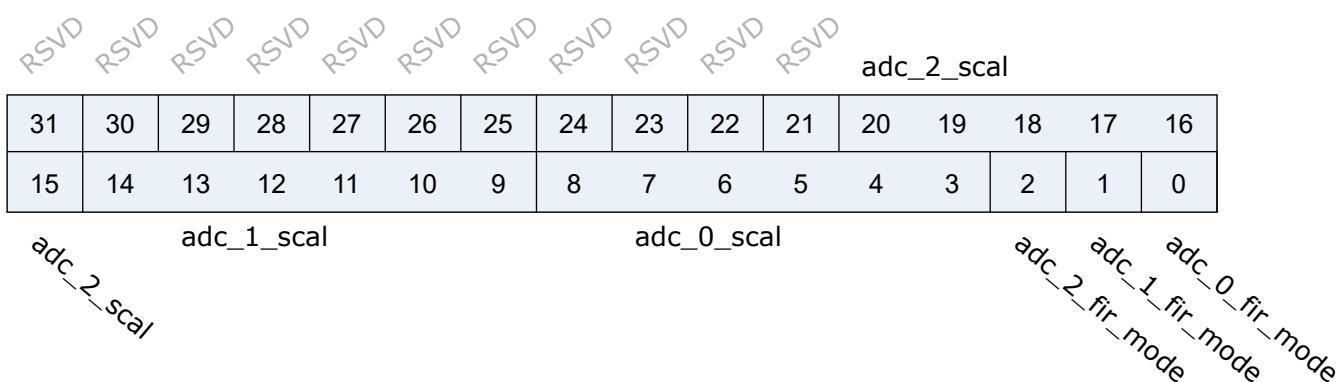


位	名称	权限	复位值	描述
31	dac_itf_en	r/w	1'd0	1:enable dac to audio dma interface
30	adc_itf_en	r/w	1'd0	1:enable adc to audio dma interface
29	aud_tx1_sel	r/w	1'd1	audio tx1 source select; 0:dac ch0, 1:dac ch1
28	aud_tx0_sel	r/w	1'd0	audio tx0 source select; 0:dac ch0, 1:dac ch1
27:25	aud_rx4_sel	r/w	3'd4	audio rx4 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
24:22	aud_rx3_sel	r/w	3'd3	audio rx3 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
21:19	aud_rx2_sel	r/w	3'd2	audio rx2 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
18:16	aud_rx1_sel	r/w	3'd1	audio rx1 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
15:13	aud_rx0_sel	r/w	3'd0	audio rx0 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
12:7	RSVD			
6	aec_1_en	r/w	1'd0	1:enable aec ch1
5	aec_0_en	r/w	1'd0	1:enable aec ch0
4	dac_1_en	r/w	1'd0	1:enable dac ch1

位	名称	权限	复位值	描述
3	dac_0_en	r/w	1'd0	1:enable dac ch0
2	adc_2_en	r/w	1'd0	1:enable adc ch2
1	adc_1_en	r/w	1'd0	1:enable adc ch1
0	adc_0_en	r/w	1'd0	1:enable adc ch0

### 21.4.3 pdm\_adc\_0

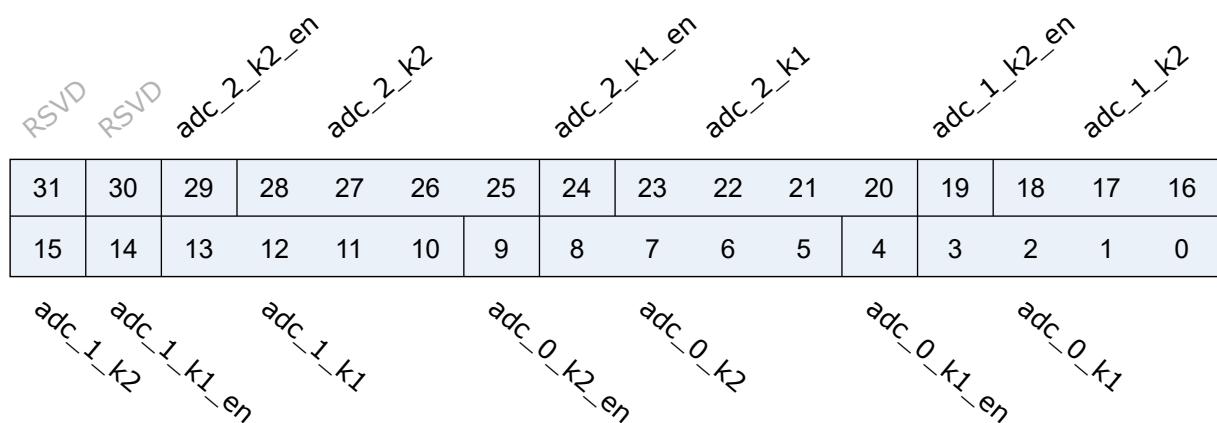
地址: 0x2000ac08



位	名称	权限	复位值	描述
31:30	adc_lfsr_mode	r/w	2'd0	0:LFSR32, 1:LFSR24, 2:LFSR16, 3:LFSR12
29	adc_dither_data	r	1'd1	read LFSR out
28:21	RSVD			
20:15	adc_2_scal	r/w	6'd32	adc ch2 scaling value; u6.5
14:9	adc_1_scal	r/w	6'd32	adc ch1 scaling value; u6.5
8:3	adc_0_scal	r/w	6'd32	adc ch0 scaling value; u6.5
2	adc_2_fir_mode	r/w	1'd0	adc fir mode
1	adc_1_fir_mode	r/w	1'd0	adc fir mode
0	adc_0_fir_mode	r/w	1'd0	adc fir mode

## 21.4.4 pdm\_adc\_1

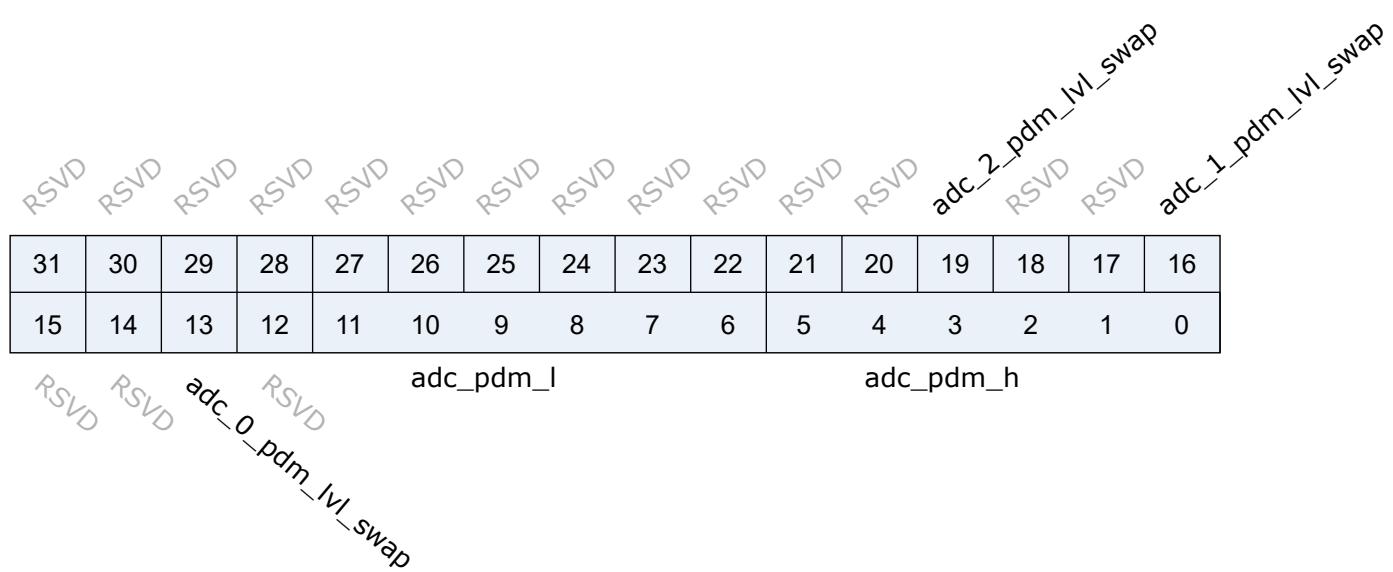
地址: 0x2000ac0c



位	名称	权限	复位值	描述
31:30	RSVD			
29	adc_2_k2_en	r/w	1'd0	adc ch2 hpf parameter k2 enable
28:25	adc_2_k2	r/w	4'd13	adc ch2 hpf parameter k2
24	adc_2_k1_en	r/w	1'd1	adc ch2 hpf parameter k1 enable
23:20	adc_2_k1	r/w	4'd8	adc ch2 hpf parameter k1
19	adc_1_k2_en	r/w	1'd0	adc ch1 hpf parameter k2 enable
18:15	adc_1_k2	r/w	4'd13	adc ch1 hpf parameter k2
14	adc_1_k1_en	r/w	1'd1	adc ch1 hpf parameter k1 enable
13:10	adc_1_k1	r/w	4'd8	adc ch1 hpf parameter k1
9	adc_0_k2_en	r/w	1'd0	adc ch0 hpf parameter k2 enable
8:5	adc_0_k2	r/w	4'd13	adc ch0 hpf parameter k2
4	adc_0_k1_en	r/w	1'd1	adc ch0 hpf parameter k1 enable
3:0	adc_0_k1	r/w	4'd8	adc ch0 hpf parameter k1

## 21.4.5 pdm\_dac\_0

地址: 0x2000ac10

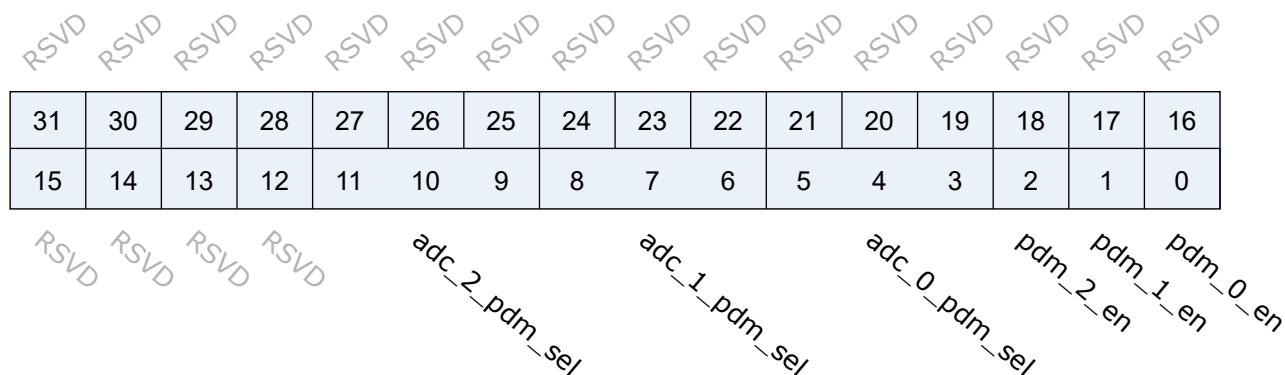


位	名称	权限	复位值	描述
31	RSVD			
30:28	mix_0_att_mode2	r/w	3'd0	0: 0db, 1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
27:25	mix_0_att_mode1	r/w	3'd0	0: 0db, 1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
24:23	mix_0_mode	r/w	2'd0	0: no mix, 1: mix second input, 2: mix sidetone/loopback
22:21	mix_0_sel	r/w	2'd0	0: 0, 1:adc ch0, 2: adc ch1, 3:adc ch2
20	adc_2_mash_bit_swap	r/w	1'd0	1:swap adc1_do1 and adc1_do2
19	adc_2_pdm_lv_swap	r/w	1'd0	1:invert pdm input data
18	adc_2_src	r/w	1'd0	0:adc, 1:pdm
17	adc_1_mash_bit_swap	r/w	1'd0	1:swap adc2_do1 and adc2_do2
16	adc_1_pdm_lv_swap	r/w	1'd0	1:invert pdm input data
15	adc_1_src	r/w	1'd0	0:adc, 1:pdm
14	adc_0_mash_bit_swap	r/w	1'd0	1:swap adc3_do1 and adc3_do2
13	adc_0_pdm_lv_swap	r/w	1'd0	1:invert pdm input data
12	adc_0_src	r/w	1'd0	0:adc, 1:pdm
11:6	adc_pdm_l	r/w	6'h3f	pdm low value
5:0	adc_pdm_h	r/w	6'h1	pdm high value

位	名称	权限	复位值	描述
-1:31	RSVD			
30:28	mix_1_att_mode2	r/w	3'd0	0: 0db, 1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
27:25	mix_1_att_mode1	r/w	3'd0	0: 0db, 1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
24:23	mix_1_mode	r/w	2'd0	0: no mix, 1: mix second input, 2: mix sidetone/loopback
22:21	mix_1_sel	r/w	2'd0	0: 0, 1:adc ch0, 2: adc ch1, 3:adc ch2
20:17	RSVD			
16:15	dac_dsm_dither_prbs_mode	r/w	1'd0	dac dsm dither Ifsr mode: 0:LFSR32, 1:LFSR24, 2:LFSR16, 3:LFSR12
14	dac_dsm_dither_en	r/w	1'd1	1:enable dac dsm dither
13:11	dac_dsm_dither_amp	r/w	3'd0	dac dsm dither amplitue
10	dac_dsm_scaling_en	r/w	1'd1	1:enable dac dsm scaling
9:6	dac_dsm_scaling_factor	r/w	4'd15	dac dsm scaling value; u4.4
5	dac_dsm_order	r/w	1'd0	0: 2-order, 1: 3-order
4:2	RSVD			
1	dac_dem_out_swap	r/w	1'd0	1:swap daidata and dacrdta
0	dac_dem_bypass	r/w	1'd0	1:bypass dac dwa
-1:14	RSVD			
13	aec_record_vld_4s_en	r/w	1'd0	0:aec record vld auto controlled by dac_rate, 1:enable aec record vld manual mode
12:11	aec_record_vld_4s_div	r/w	2'd0	aec record vld manual divide rate
10:8	aec_1_atten_mode	r/w	3'd0	aec ch1 attenuation mode: 0:no attenuation, 1:drop 1LSB, 2:drop 2LSB, 3:drop 3LSB, 4:drop 6LSB, 5:drop 9LSB, 6:drop 12LSB
7:5	aec_0_atten_mode	r/w	3'd0	aec ch0 attenuation mode: 0:no attenuation, 1:drop 1LSB, 2:drop 2LSB, 3:drop 3LSB, 4:drop 6LSB, 5:drop 9LSB, 6:drop 12LSB
4:0	RSVD			

## 21.4.6 pdm\_pdm\_0

地址: 0x2000ac1c



位	名称	权限	复位值	描述
31:12	RSVD			
11:9	adc_2_pdm_sel	r/w	3'd2	adc ch2 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
8:6	adc_1_pdm_sel	r/w	3'd1	adc ch1 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
5:3	adc_0_pdm_sel	r/w	3'd0	adc ch0 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
2	pdm_2_en	r/w	1'd0	1:enable pdm_2
1	pdm_1_en	r/w	1'd0	1:enable pdm_1
0	pdm_0_en	r/w	1'd0	1:enable pdm_0

## 21.4.7 pdm\_rsvd0

地址: 0x2000ac20

rsvd0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

rsvd0

位	名称	权限	复位值	描述
31:0	rsvd0	r/w	32'hffff	

#### 21.4.8 pdm\_dbg\_0

地址: 0x2000ac24

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:30	RSVD			
29:24	aud_test_read_sel	r/w	6'd0	select aud_test_read(0x28) test point
23	adc_test_din_en	r/w	1'd0	1:enable adc test data input from GPIO
22	dac_test_din_en	r/w	1'd0	1:enable dac test data input from GPIO
21	adc_test_clkin_en	r/w	1'd0	1:enable adc test clck input from GPIO
20	dac_test_clkin_en	r/w	1'd0	1:enable dac test clck input from GPIO
19:18	audio_test_out_sel	r/w	2'd0	audio test data to GPIO select: 0:no data, 1:adc ch0/1/2, 2:dac ch0 dwa, 2:dac ch1 dwa
17:4	RSVD			
3:1	aud_sin_step	r/w	3'd2	step of dac audio sin table @FS=192k 0:div1, 1:div2, 2:div4, 3:div6, 4:div8, 5:div12, 6:div24
0	aud_sin_en	r/w	1'd0	1:enable audio dac sin generator

#### 21.4.9 pdm\_dbg\_1

地址: 0x2000ac28

aud_test_read															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

aud\_test\_read

位	名称	权限	复位值	描述
31:0	aud_test_read	r	32'd0	audio test read value

#### 21.4.10 pdm\_dbg\_2

地址: 0x2000ac2c

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
adc_fir_4s_val															

位	名称	权限	复位值	描述
31:26	RSVD			
25	adc_in_2_test_sel	r/w	1'd0	adc ch2 test data select; 1:from adc sin generator, 0:from fir force value
24	adc_in_1_test_sel	r/w	1'd0	adc ch1 test data select; 1:from adc sin generator, 0:from fir force value
23	adc_in_0_test_sel	r/w	1'd0	adc ch0 test data select; 1:from adc sin generator, 0:from fir force value
22	adc_2_fir_4s_en	r/w	1'd0	1:force adc ch2 fir output
21	adc_1_fir_4s_en	r/w	1'd0	1:force adc ch1 fir output
20	adc_0_fir_4s_en	r/w	1'd0	1:force adc ch0 fir output
19:0	adc_fir_4s_val	r/w	20'd0	force value of adc fir output

#### 21.4.11 pdm\_dbg\_3

地址: 0x2000ac30

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

位	名称	权限	复位值	描述
31:24	RSVD			
23	dac_in_1_test_sel	r/w	1'd0	dac ch1 test data select; 0:from dac sin generator, 1:from dac force value
22	dac_in_0_test_sel	r/w	1'd0	dac ch0 test data select; 0:from dac sin generator, 1:from dac force value
21	dac_dwa_1_4s_en	r/w	1'd0	1:force dac ch1 dwa data from dac_4s_val[6:0]
20	dac_dwa_0_4s_en	r/w	1'd0	1:force dac ch0 dwa data from dac_4s_val[6:0]
19:0	dac_4s_val	r/w	20'd0	dac force value

#### 21.4.12 pdm\_dbg\_4

地址: 0x2000ac34

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD										
pdm_in_ratio_4s_val	adc_in_ch0_ratio_4s_val	adc_in_ch1_ratio_4s_val	adc_in_ch2_ratio_4s_val	adc_in_ratio_4s_val	adc_in_ratio_4s_val	RSVD									

位	名称	权限	复位值	描述
31:11	RSVD			
10:8	aec_fs_rate_4s_val	r/w	3'd0	
7:6	dac_out_ratio_4s_val	r/w	2'd0	
5	pdm_in_ratio_4s	r/w	1'd0	
4	pdm_in_ratio_4s_val	r/w	1'd0	
3	adc_in_ch2_ratio_4s_val	r/w	1'd0	
2	adc_in_ch1_ratio_4s_val	r/w	1'd0	
1	adc_in_ch0_ratio_4s_val	r/w	1'd0	
0	adc_in_ratio_4s_val	r/w	1'd0	

### 21.4.13 pdm\_adc\_s0

地址: 0x2000ac38

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

adc\_s0\_volume

位	名称	权限	复位值	描述
31:9	RSVD			
8:0	adc_s0_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

### 21.4.14 pdm\_adc\_s1

地址: 0x2000ac3c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

adc\_s1\_volume

位	名称	权限	复位值	描述
31:9	RSVD			
8:0	adc_s1_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

### 21.4.15 pdm\_adc\_s2

地址: 0x2000ac40

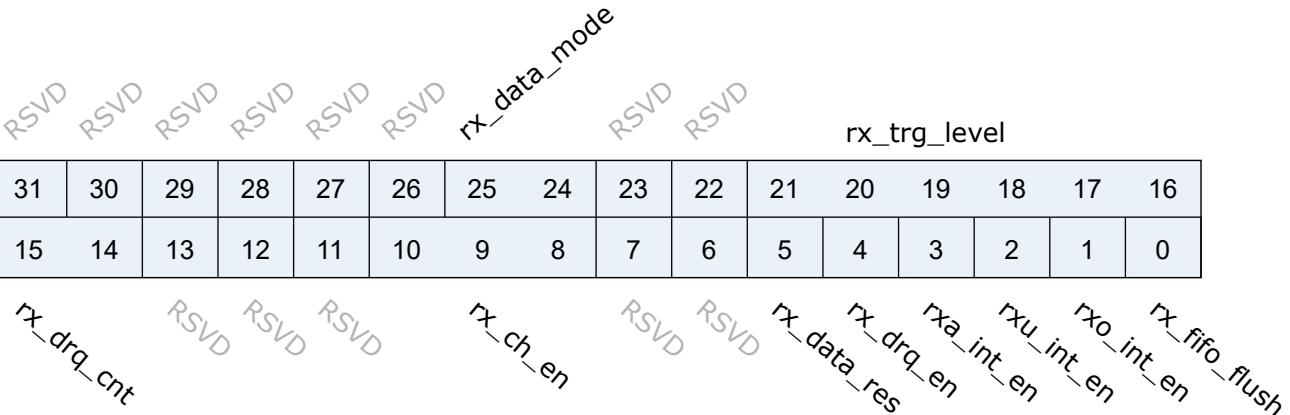
RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

adc\_s2\_volume

位	名称	权限	复位值	描述
31:9	RSVD			
8:0	adc_s2_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

#### 21.4.16 pdm\_rx\_fifo\_ctrl

地址: 0x2000ac80



位	名称	权限	复位值	描述
31:26	RSVD			

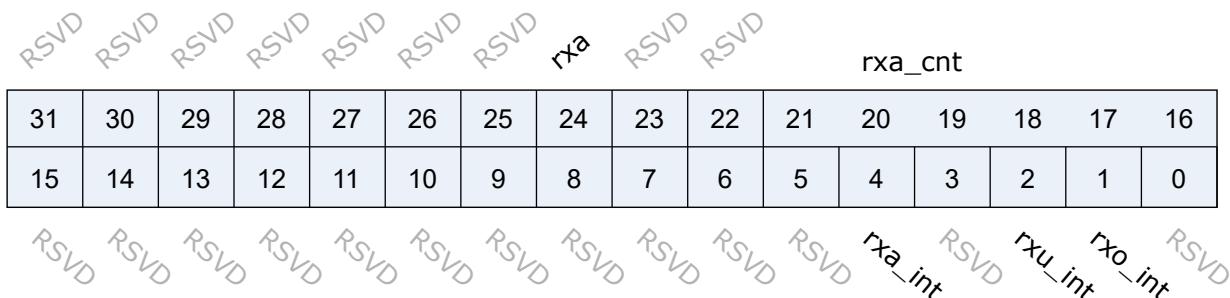
位	名称	权限	复位值	描述
25:24	rx_data_mode	r/w	2'b0	<p>RX_FIFO_DATOUT_MODE.</p> <p>RX FIFO DATA Output Mode (Mode 0, 1, 2, 3)</p> <p>Mode 0: Valid data's MSB is at [31] of RX_FIFO register</p> <p>Mode 1: Valid data's MSB is at [23] of RX_FIFO register</p> <p>Mode 2: Valid data's MSB is at [19] of RX_FIFO register</p> <p>Mode 3: Valid data's MSB is at [15] of RX_FIFO register</p> <p>Note: Expanding '0' at LSB of RX FIFO register (data invalid region)</p> <p>Expanding sign bit at MSB of RX FIFO register (data invalid region)</p> <p>For 20-bit received audio sample resolution:</p> <p>Mode 0: RXDATA[31:0] = FIFO_O[19:0], 12'h0</p> <p>Mode 1: RXDATA[31:0] = 8FIFO_O[19], FIFO_O[19:0], 4'h0</p> <p>Mode 2: RXDATA[31:0] = 12FIFO_O[19], FIFO_O[19:0]</p> <p>Mode 3: RXDATA[31:0] = 16FIFO_O[19], FIFO_O[19:4]</p> <p>For 16-bit received audio sample resolution:</p> <p>Mode 0: RXDATA[31:0] = FIFO_O[19:4], 16'h0</p> <p>Mode 1: RXDATA[31:0] = 8FIFO_O[19], FIFO_O[19:4], 8'h0</p> <p>Mode 2: RXDATA[31:0] = 12FIFO_O[19], FIFO_O[19:4], 4'h0</p> <p>Mode 3: RXDATA[31:0] = 16FIFO_O[19], FIFO_O[19:4]</p>
23:22	RSVD			

位	名称	权限	复位值	描述
21:16	rx_trg_level	r/w	6'd23	<p>RX_FIFO_TRIGGER_LEVEL.</p> <p>RX FIFO Trigger Level (RXTL[5:0])</p> <p>Interrupt and DMA request trigger level for RX FIFO Data Available condition</p> <p>IRQ/DRQ Generated when WLEVEL &gt; RXTL[5:0]</p> <p>Notes:</p> <p>WLEVEL represents the number of valid samples in the RX FIFO</p>
15:14	rx_drq_cnt	r/w	2'b0	<p>RX_DRQ_CLR_CNT.</p> <p>When RX FIFO available data less than or equal N, DRQ Request will be de-asserted. N is defined here:</p> <ul style="list-style-type: none"> <li>00: IRQ/DRQ de-asserted when WLEVEL &lt;= RXTL[5:0]</li> <li>01: IRQ/DRQ de-asserted when WLEVEL &lt; 8</li> <li>10: IRQ/DRQ de-asserted when WLEVEL &lt; 16</li> <li>11: IRQ/DRQ de-asserted when WLEVEL &lt; 32</li> </ul> <p>WLEVEL represents the number of valid samples in the RX FIFO</p>
13:11	RSVD			
10:8	rx_ch_en	r/w	3'b0	<p>RX_FIFO_DATIN_SRC.</p> <p>RX FIFO Data Input Source Select.</p> <p>0: Disable 1: Enable</p> <p>Bit10: ADC3 data</p> <p>Bit9: ADC2 data</p> <p>Bit8: ADC1 data</p> <p>When some of the above bits set to '1', these data are always arranged in order from low-bit to high-bit.(bit8-&gt;bit10)</p>
7:6	RSVD			
5	rx_data_res	r/w	1'b0	<p>RX_SAMPLE_BITS.</p> <p>Receiving Audio Sample Resolution</p> <p>0: 16 bits</p> <p>1: 20 bits</p>

位	名称	权限	复位值	描述
4	rx_drq_en	r/w	1'b0	ADC_DRQ_EN. ADC FIFO Data Available DRQ Enable. 0: Disable 1: Enable
3	rx_a_int_en	r/w	1'b0	ADC_IRQ_EN. ADC FIFO Data Available IRQ Enable. 0: Disable 1: Enable
2	rx_u_int_en	r/w	1'b0	ADC_UNDERRUN_IRQ_EN. ADC FIFO Under Run IRQ Enable 0: Disable 1: Enable
1	rx_o_int_en	r/w	1'b0	ADC_OVERRUN_IRQ_EN. ADC FIFO Over Run IRQ Enable 0: Disable 1: Enable
0	rx_fifo_flush	w1c	1'b0	ADC_FIFO_FLUSH. ADC FIFO Flush. Write ‘1’ to flush TX FIFO, self clear to ‘0’

#### 21.4.17 pdm\_rx\_fifo\_status

地址: 0x2000ac84



位	名称	权限	复位值	描述
31:25	RSVD			

位	名称	权限	复位值	描述
24	rxa	r	1'b0	RXA. RX FIFO Available 0: No available data in RX FIFO 1: More than one sample in RX FIFO (>= 1 word)
23:22	RSVD			
21:16	rxa_cnt	r	6'h0	RXA_CNT. RX FIFO Available Sample Word Counter
15:5	RSVD			
4	rxa_int	r	1'b0	RXA_INT. RX FIFO Data Available Pending Interrupt 0: No Pending IRQ 1: Data Available Pending IRQ Automatic clear if interrupt condition fails.
3	RSVD			
2	rxu_int	r	1'b0	RXU_INT. RX FIFO Underrun Pending Interrupt 0: No Pending IRQ 1: FIFO Underrun Pending IRQ Write ‘1’ to clear this interrupt
1	rxo_int	r	1'b0	RXO_INT. RX FIFO Overrun Pending Interrupt 0: No Pending IRQ 1: FIFO Overrun Pending IRQ Write ‘1’ to clear this interrupt
0	RSVD			

#### 21.4.18 pdm\_rx\_fifo\_data

地址: 0x2000ac88

rx\_data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

rx\_data

位	名称	权限	复位值	描述
31:0	rx_data	r	32'h0	<p>RX_DATA. RX Sample</p> <p>Host can get one sample by reading this register. The left channel sample data is first and then the right channel sample.</p>

## 22.1 简介

芯片内置一个 AUDIO 音频处理模块，支持 pdm/模拟接口麦克风录音，立体声放音。

## 22.2 主要特征

- 集成 3 路 20bit ADC, 可支持 3 路模拟 mic 差分输入
  - 采样率: 8k~96k
  - 信噪比 (A-W): 97dB @ 0dB 增益
  - 谐波失真 + 噪声: -87dB @ 0dB 增益
  - 模拟前置增益: 0dB, 6~42 dB, 3dB 一档
- 一个电压可设置的低噪声偏置电源给模拟 MIC 供电，调节范围 1.8V~2.5V
- 可调节的高通滤波器和独立的数字音量控制，用于 ADC 通路
- 一个 5 段的参数均衡器，用于 ADC 通路
- 支持数字 mic 接口，复用输入 GPIO
- 集成 2 路 20bit DAC, 可支持 2 路模拟 LINEOUT 差分输出，输出具有斜坡上电功能
  - 采样率: 8k~192k
  - 信噪比 (A-W): 103dB @ 0dB 增益
  - 谐波失真 + 噪声: -90dB @ 0dB 增益
  - 模拟输出增益: 0dB, 6~42 dB, 3dB 一档
- 独立的数字音量控制，并可以支持音量柔和调节/静音，用于 DAC 播放通路
- 支持动态范围控制，灵活调节的 10 段参数均衡器，用于 DAC 播放通路
- 支持语音活动检测功能，可用于低功耗的语音唤醒应用
- 32 位宽度的发送、接受 FIFO
- 支持 DMA 传输模式

## 22.3 功能描述

AUDIO 模块基本框图如图所示。

AUDIO 模块包含了三路 ADC 模数转换器，经过 PGA 增益后，经过抽取滤波器，HPF 滤波器之后，进入音量控制模块。用户可以通过音量控制模块，控制静音/非静音，控制音量大小，控制声音渐入/渐出效果。五段 EQ 滤波器会优化音频性能，对音频各个频段进行增益和衰减，从而优化录音音色。AGC 会根据音频数据的幅值对音量进行调整，从而优化录音的性能，减少出现录音的顶部失真。VAD 模块是内部自带的唤醒模块，VAD 模块会区分环境中的噪声信号与有用的音频信号，当侦测到有效的音频信号输入时，会产生一个 VAD\_INT 事件。录音的数据最终会存放在深度为 32 的 FIFO 中，可以由 RX\_FIFO\_CTRL[25:24] 寄存器来控制 FIFO 的存放格式。RX\_FIFO\_CTRL[5] 配置采样的分辨率。录音的数据接口可以选择为 PDM 数字接口，或者模拟麦克风接口，可以由 DAC\_0[12] DAC\_0[15] DAC\_0[18] 位来控制接口的类型。

AUDIO 模块包含了两路 DAC 数模转换器，数据从 FIFO 中取出，经过混声器，以及 EQ 滤波器，音量控制，插入滤波器，最终调制为模拟信号，驱动喇叭实现播放。混声器可以将 FIFO 里面读出的音频数据与 ADC 通路录音数据进行混合。EQ 滤波器对各个频段进行增益衰减，从而对播放音频的音色进行修改。音量控制模块会对播放音频的音量以及键入渐出方式进行控制。

### 22.3.1 AUDIO 中断

AUDIO 有着丰富的中断控制，包括以下几种中断模式：

- TX FIFO 请求中断
- TX FIFO underrun 中断
- TX FIFO overrun 中断
- RX FIFO 请求中断
- RX FIFO underrun 中断
- RX FIFO overrun 中断
- 录音音量调节中断
- 播放音量调节中断
- VAD 中断

当 TX\_FIFO\_CTRL 中 TX\_DRQ\_CNT 大于 TX\_TRG\_LEVEL 时，产生 TX FIFO 请求中断。当条件不满足时该中断标志会自动清除。

当 TX FIFO 中并没有数据，但是用户却通过 TX\_FIFO\_CTRL 中的 TX\_CH\_EN 使能了 TX FIFO 调制，则会进入 TX FIFO underrun 中断。

当用户填入超过 TX FIFO 最大深度的数据的时候，会导致 TX FIFO 溢出，从而产生 TX FIFO overrun 中断

当 RX\_FIFO\_CTRL 中 RX\_DRQ\_CNT 大于 RX\_TRG\_LEVEL 时，产生 RX FIFO 请求中断。当条件不满足时该中断标志会自动清除。

当 RX FIFO 中并没有数据，但是用户却通过 RX\_FIFO\_CTRL 中的 RX\_CH\_EN 使能了 RX FIFO 调制，则会进入 RX FIFO underrun 中断。

当用户填入超过 RX FIFO 最大深度的数据的时候，会导致 RX FIFO 溢出，从而产生 RX FIFO overrun 中断

当 ADC 的录音音量调节完成的时候，会触发 ADC 音量调节完成中断，对于一般的应用场景而言，ADC 的音量调节是立即完成的。

当 DAC 的播放音量调节完成的时候，会触发 DAC 音量调节完成中断，因为 DAC 播放会有渐变的效果，所以音量调节不一定是立即完成的，会根据 DAC\_H\_CTRL\_MODE 相关寄存器来确定调节时间。

### 22.3.2 FIFO 格式控制

TX\_FIFO\_CTRL 与 RX\_FIFO\_CTRL(RX\_FIFO\_CTRL 与 TX\_FIFO\_CTRL 完全一致，故以下简称为 FIFO\_CTRL) 可以控制音频数据储存在 FIFO 的格式。

用户通过配置 FIFO\_CTRL[5] 来选择音频的分辨率。

当分辨率选择位 16bits 时，FIFO 控制器支持如下四种数据存储格式，由 FIFO\_CTRL[25:24] 来决定。

- Mode 0:
  - DATA[31:0] = {FIFO[19:14], 16'h0}
- Mode 1:
  - DATA[31:0] = {8{FIFO[19]}, FIFO[19:4], 8'h0}
- Mode 2:
  - DATA[31:0] = {12{FIFO[19]}, FIFO[19:4], 4'h0}
- Mode 3:
  - DATA[31:0] = {16{FIFO[19]}, FIFO[19:4]}

在录音/播放的时候，32Bits 数据的转换结果或待解调的数字音源在左边，记作 DATA[31:0]。他将被储存在 FIFO 中的格式如右边，因为录音的分辨率为 20bits，所以当选择分辨率为 16bits 的时候，需要对 20bit 的分辨率做一些剪裁，因此选取了 20bit 分辨率的高 16bits 作为最终的结果 FIFO[19:14]，并把它保存在了高 16bits 的位置，低 16bits 采用用 0 补齐的操作。Mode1、2、3 与 Mode0 的表示方式相同，值得说明的是，8{FIFO[19]} 符号表示的是会用 bit[19] 的值来填充高八位。

因此，当选择分辨率为 16bits 的时候，AUDIO 提供了四种模式来存放转换/输出的数字结果。

当分辨率为 20bits 的时候，四种模式的储存方式如下

- Mode 0:
  - DATA[31:0] = {FIFO[19:0], 12'h0}
- Mode 1:
  - DATA[31:0] = {8{FIFO[19]}, FIFO[19:0], 4'h0}
- Mode 2:
  - DATA[31:0] = {12{FIFO[19]}, FIFO[19:0]}
- Mode 3:
  - DATA[31:0] = {16{FIFO[19]}, FIFO[19:4]}

最高有效位的分布

- Mode 0:
  - 有效数据的最高位在 31 bits
- Mode 1:
  - 有效数据的最高位在 23 bits

- Mode 2:
  - 有效数据的最高位在 19 bits
- Mode 3:
  - 有效数据的最高位在 15 bits

### 22.3.3 FIFO 的启动与 DMA 搬运

AUDIO 的 FIFO 数据可以通过 DMA 进行搬运。

用户可以通过 RX\_FIFO\_STATUS/TX\_FIFO\_STATUS 寄存器实时获得目前 FIFO 有效数据的数量。

通过配置 FIFO\_CTRL[15:14] 来选择发起 DMA request 的 FIFO count 阈值，是 8/16/32，或者是由 FIFO\_CTRL[22:16] 配置来决定。

当 count 的值大于设定阈值，并且 TX\_FIFO\_CTRL[9:8] 或 RX\_FIFO\_CTRL[12:8] 对应通路的 FIFO 被使能，则会发起一次 DMA 搬运。

注意，启动 TX FIFO 时，如果 TX FIFO 里面并没有有效的数据，则会触发 tx underrun 错误。因此要注意软件配置顺序。

## 23.1 简介

芯片搭载 PSRAM 控制器，可以驱动多种型号的 PSRAM。

## 23.2 主要特征

- 最大支持 PSRAM 时钟为 200MHz
- 支持 linear brust/Warp brust 两种模式选择

## 23.3 功能描述

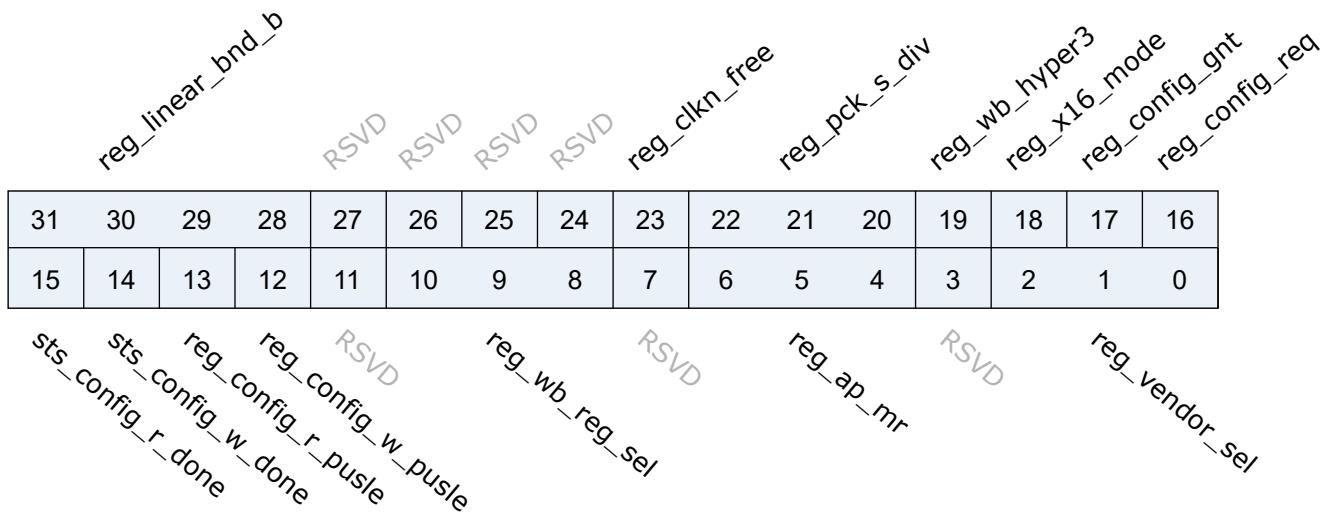
PSRAM 控制器可以实现对多种 PSRAM 的内部寄存器读写，以及 memory 读写。

## 23.4 寄存器描述

名称	描述
psram_configure	

### 23.4.1 psram\_configure

地址: 0x20052000



位	名称	权限	复位值	描述
31:28	reg_linear_bnd_b	r/w	4'd10	Linear burst boundary shift bit
27:24	RSVD			
23	reg_clkn_free	r/w	1'b1	Clock# free run
22:20	reg_pck_s_div	r/w	3'd0	EMI AXI bus clock division from psram_ck
19	reg_wb_hyper3	r/w	1'b0	Winbond Hyper3 bus
18	reg_x16_mode	r/w	1'b0	16-bit pSRAM mode enable
17	reg_config_gnt	r	1'b0	pSRAM register configure grant
16	reg_config_req	r/w	1'b0	pSRAM register configure request
15	sts_config_r_done	r	1'b1	pSRAM register read done
14	sts_config_w_done	r	1'b1	pSRAM register write done
13	reg_config_r_pusle	w1p	1'b0	pSRAM configuration read enable
12	reg_config_w_pusle	w1p	1'b0	pSRAM configuration write enable
11	RSVD			
10:8	reg_wb_reg_sel	r/w	2'd0	Winbond pSRAM Register R/W selection 3'd0 - ID0 / 3'd1 - ID1 3'd2 - CR0 / 3'd3 - CR1 / 3'd4 - CR2 3'd5 - CR3 / 3'd6 - CR4
7	RSVD			

位	名称	权限	复位值	描述
6:4	reg_ap_mr	r/w	3'd0	APMemory pSRAM Mode Register R/W selection 3'd0=MA0 / 3'd1=MA1 / 3'd2=MA2 / 3'd3=MA3 / 3'd4=MA4 / 3'd6=MA6 / 3'd7=MA8
3	RSVD			
2:0	reg_vendor_sel	r/w	3'b010	pSRAM vendor selection [0] - Winbond [1] - APM_XCELLA [2] - APM_HYPER

## 24.1 简介

EMAC 模块是一个兼容 IEEE 802.3 的 10/100Mbps 以太网 MAC(Ethernet Media Access Controller)。其包含状态及控制寄存器组，收发模块，收发缓冲描述符组，主机接口，MDIO 接口，物理层芯片 (PHY) 接口。

状态及控制寄存器组包含了 EMAC 的状态位及控制位，是与用户程序的接口，负责控制数据收发，并汇报状态。

收发模块负责根据收发描述符内的控制字，从指定内存位置取得数据帧，添加前导，CRC，并扩充短的帧后通过 PHY 发出；或是从 PHY 接收数据，并根据收发缓冲描述符，将数据放入指定内存。收发完成后设置相关的事件标志。如果使能了事件中断，将通过中断请求到主机进行处理。

MDIO 及 MII/RMII 接口负责与 PHY 进行通信，包括读写 PHY 的寄存器，以及数据包的收发。

## 24.2 主要特征

- 兼容 IEEE 802.3 定义的 MAC 层功能
- 支持 IEEE 802.3 定义的 MII/RMII 接口的 PHY
- 通过 MDIO 接口与 PHY 交互
- 支持 10Mbps 与 100Mbps 以太网
- 支持半双工与全双工
- 在全双工模式下，支持自动流控及生成控制帧
- 在半双工模式下，支持碰撞检测及重传
- 支持 CRC 的生成及校验
- 数据帧前导生成及移除
- 发送时，自动扩展短的数据帧
- 检测过长或过短的数据帧 (长度限制)
- 可传输长数据帧 (> 标准以太帧长度)
- 自动丢弃重发次数超限或帧间隙过小的数据包
- 广播包过滤

- 用于保存多达 128 个 BD(Buffer Descriptor) 的内部 RAM
- 在发送时，支持将一个数据包分拆配置到多个连续的 BD
- 发送/接收的各种事件标志
- 在事件发生时产生对应中断

## 24.3 功能描述

EMAC 模块的组成如下图。

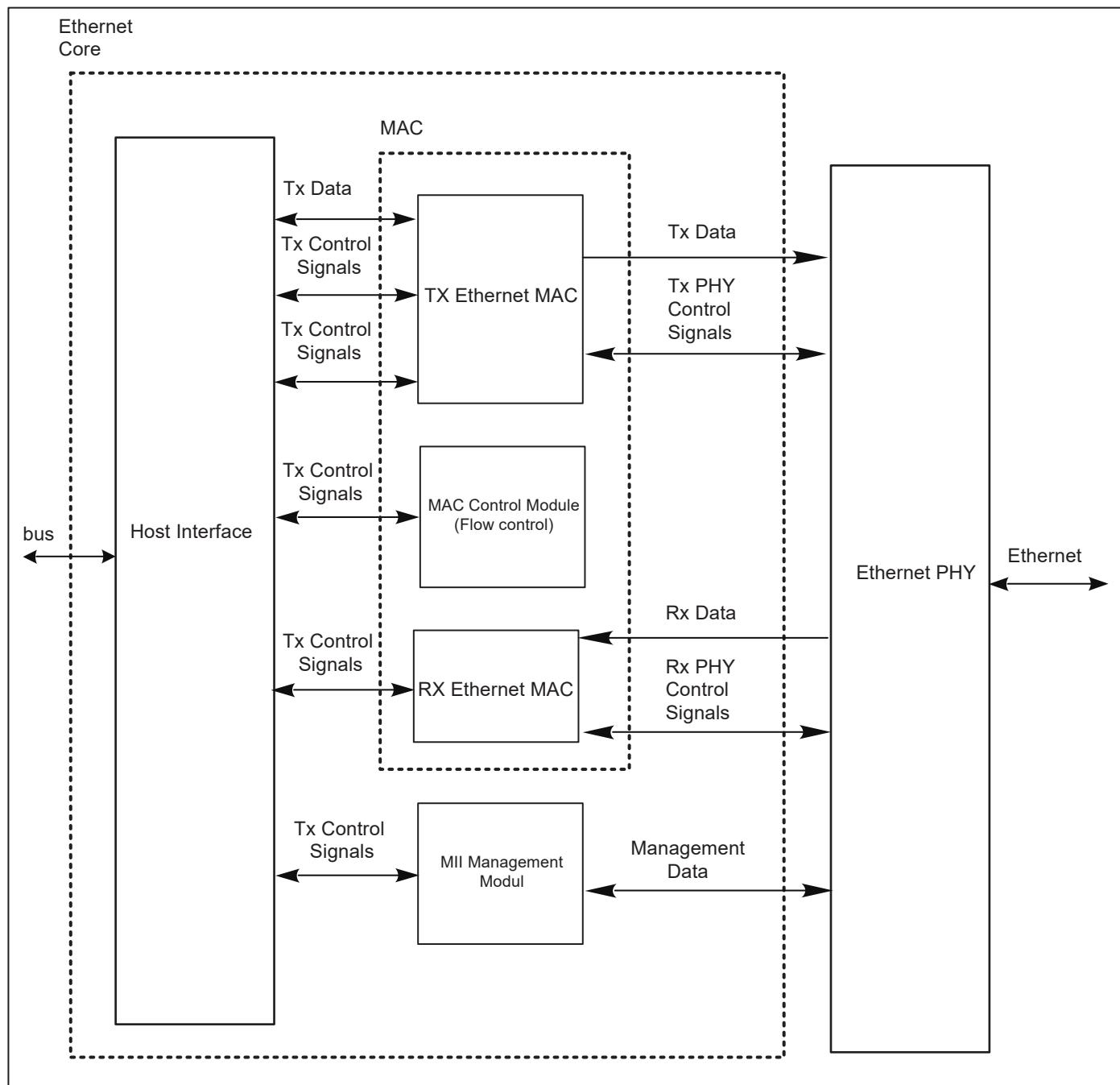


图 24.1: EMAC 框图

模块的控制寄存器通过 MDIO 接口，可以读写 PHY 的寄存器，从而实现配置、选择模式(半/全双工)、发起协商等操作。接收模块过滤并检查收到的数据帧：是否有合法的前导，FCS，长度等，并根据缓冲描述符(BD)，将数据存放到指定内存地址。发送模块根据数据缓冲描述符，从内存中取得数据，添加前导，FCS，pad 等，然后根据 CSMA/CD 协议，将数据发出。如果检测到 CRS，将会延迟重试。收发缓冲描述符组连接到系统 RAM，此 RAM 用于保存发送和接收的以太网数据帧。每个描述符包含相应的控制状态字以及对应的缓冲内存地址。描述符一共有 128 组，用于发送或者接收，可以灵活分配。

## 24.4 时钟

EMAC 模块需要一路时钟用于同步收发 (100Mbps 时，25MHz(MII) 或 50MHz(RMII); 10Mbps 时，2.5MHz)。此时钟必须在 EMAC 与 PHY 之间同步。

## 24.5 收发缓冲描述符 (BD, Buffer Descriptor)

收发缓冲描述符，用于提供 EAMC 与数据帧缓存地址信息之间的关联，对收发数据帧进行控制，以及提供收发状态提示。每个描述符由两个连续的 word(32bit) 构成，低地址的 word 提供了本 buffer 包含的数据帧的长度，控制及状态位；高地址的 word 是内存指针。具体的 BD 描述可以参考寄存器描述章节。需要注意的是：对于 BD，需要按 word 写入。EMAC 模块支持 128 个 BD，由发送/接收逻辑共享，可自由组合。但发送 BD 总是占据前面的连续区域(个数由 MAC\_TX\_BD\_NUM 寄存器中的 TXBDNUM 域来指定)。EMAC 按照 BD 的顺序，循环处理发送/接收 BD，直到遇到标记为 WR 的 BD 就回绕到发送/接收各自的首个 BD。

## 24.6 PHY 交互

PHY 交互寄存器组提供了与 PHY 交互需要的命令及数据通信的方式。EMAC 通过 MDIO 接口控制 PHY 的工作模式，并保证两者的工作模式匹配(速率，全/半双工等)。数据包通过 MII/RMII 接口在 EMAC 与 PHY 之间交互，可以通过 EMAC 的模式寄存器(EMAC\_MODE) 中的 RMII\_EN 位选择：当此 bit 为 1，则选择 RMII 模式，否则就是 MII 模式。MII 及 RMII 模式均支持 IEEE 802.3u 标准中指定的 10Mbps 与 100Mbps 的传输速率。MII 及 RMII 的传输信号描述与下表。

表 24.1: 传输信号

名称	MII	RMII
EXTCK_EREFCK	ETXCK: 发送时钟信号	EREFCK: 参考时钟
ECRS	ECRS: 载波探测	-
ECOL	ECOL: 碰撞检测	-
ERXDV	ERXDV: 数据 valid	ECRSDV: 载波检测/数据 valid
ERX0-ERX3	ERX0-ERX3: 4-bit 接收数据	ERX0-ERX1: 2-bit 接收数据
ERXER	ERXER: 接收错误指示	ERXER: 接收错误指示
ERXCK	ERXCK: 接收时钟信号	-
ETXEN	ETXEN: 发送使能	ETXEN: 发送使能
ETX0-ETX3	ETX0-ETX3: 4-bit 发送数据	ETX0-ETX1: 2-bit 发送数据
ETXER	ETXER: 发送错误指示	-
EMDC	MDIO Clock	MDIO Clock
EMDIO	MDIO Data Input Output	MDIO Data Input Output

RMII 接口引脚较少，使用 2-bit 数据线用于收发，在 100Mbps 速率时，需要提供 50MHz 的参考时钟。

## 24.7 编程流程

### 24.7.1 PHY 初始化

- 根据 PHY 类型，设置 EMAC\_MODE 寄存器中的 RMII\_EN 位来选择合适的连接方式
- 设置 EMAC 的 MAC 地址到 EMAC\_MAC\_ADDR0 与 EMAC\_MAC\_ADDR1 中
- 通过编程 EMAC\_MIIMODE 寄存器中的域 CLKDIV，为 MDIO 部分设置合适的时钟
- 设置对应 PHY 的地址到寄存器 EMAC\_MIIADDRESS 的域 FIAD 中
- 根据 PHY 的手册，通过 EMAC\_MIICOMMAND 与 EMAC\_MIITX\_DATA 寄存器发送命令
- 读取 PHY 的数据会保存在 EMAC\_MIIRX\_DATA 寄存器中
- 通过 EMAC\_MIISTATUS 寄存器可以查询与 PHY 命令交互的状态

基础的交互完成后，应当使 PHY 进入自动协商状态。协商完成之后，根据协商结果编程模式到 EMAC\_MODE 寄存器中的 FULLD 位。

### 24.7.2 发送数据帧

- 配置 EMAC\_MODE 寄存器中数据帧格式、间隔等位域
- 通过配置 EMAC\_TX\_BD\_NUM 寄存器中的 TXBDNUM 域来指定发送所使用的 BD 的个数，那么剩余的就是 RX 的 BD
- 在内存中准备好需要发送的数据帧
- 将数据帧的地址填写到对应发送 BD 的数据指针域 (word 1) 中
- 清空对应发送 BD 的控制与状态域 (word 0) 中的状态标记，并设置控制域 (CRC 使能，PAD 使能，中断使能等)
- 写入数据帧长度，并设置好 RD 域，告知 EMAC 此 BD 数据需要发送；如需要，设置上 IRQ 位，以使能中断
- 特别的，如果是最后一个发送的 BD，需要设置上 WR 位，EMAC 会在处理完这个 BD 之后“回绕”到第一个发送 BD 进行处理
- 如果有多个 BD 需要发送，则重复设置 BD 的步骤以填充所有的发送 BD
- 如果一个数据包只包含在一个 BD 中，那么需要设置其 EOF 位为 1
- 如果一个数据包分在多个 BD 里进行发送，那么只需要将其占用的最后一个 BD 标记为数据包结束 (设置 EOF 位)
- 如果需要使能发送中断，还需要配置 EMAC\_INT\_MASK 寄存器中的 TX 相关位
- 配置 EMAC\_MODE 寄存器中的 TXEN 位，以使能发送
- 如果使能了中断，在发送的中断中，可用通过 EMAC\_TX\_BD\_NUM 寄存器中的 TXBDNUM 域获取当前的 BD
- 根据当前 BD 的状态字进行相应的处理
- 数据已被发送出去的 BD，其控制域中的 RD 位会被硬件清零，且不会被再次发送；需要填充新数据后，置位 RD，此 BD 即可再次用于发送

### 24.7.3 接收数据帧

- 配置 EMAC\_MODE 寄存器中数据帧格式、间隔等位域
- 通过配置 EMAC\_TX\_BD\_NUM 寄存器中的 TXBDNUM 域来指定发送所使用的 BD 的个数，那么剩余的就是 RX 的 BD
- 在内存中准备好接收数据的区域
- 将数据帧的地址填写到对应接收 BD 的数据指针域 (word 1) 中
- 清空对应发送 BD 的控制与状态域 (word 0) 中的状态标记，并设置控制域 (中断使能等)
- 写入可接收的数据帧长度，并设置好 E 位域，告知 EMAC 此 BD 空闲，可以用于数据接收；如需要，设置上 IRQ 位，以使能中断
- 特别的，如果是最后一个有效接收 BD，需要设置上 WR 位，EMAC 会在处理完这个 BD 之后“回绕”到第一个接收 BD 进行处理
- 如果有多个 BD 可供接收数据，则重复设置 BD 的步骤以填充所有的 BD
- 如果需要使能接收中断，还需要配置 EMAC\_INT\_MASK 寄存器中的 RX 相关位
- 配置 EMAC\_MODE 寄存器中的 RXEN 位，以使能接收
- 如果使能了中断，在接收的中断中，可用通过 EMAC\_TX\_BD\_NUM 寄存器中的 RXBDNUM 域获取当前的 BD
- 根据当前 BD 的状态字进行相应的处理
- 接收完成的 BD，其控制域中的 E 位会被硬件清零，且不会被再次用于接收；需要取走数据，置位 E，此 BD 即可再次用于接收

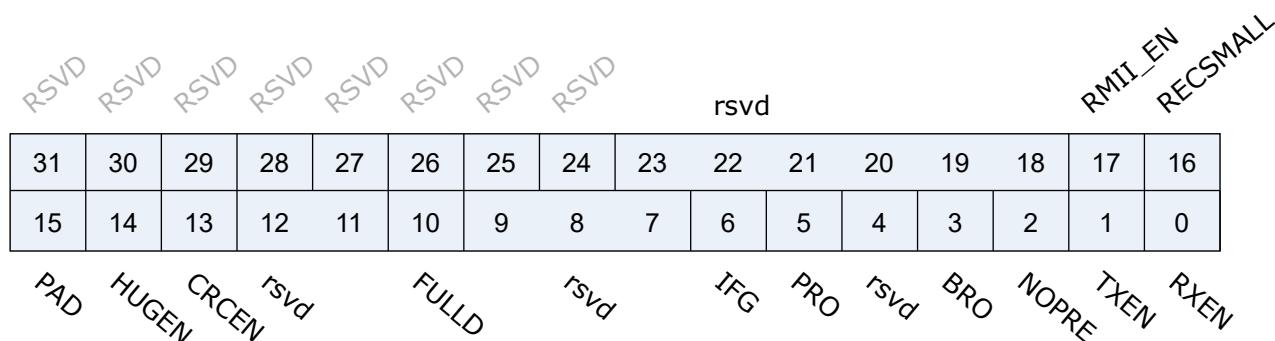
## 24.8 寄存器描述

名称	描述
MODE	
INT_SOURCE	
INT_MASK	
IPGT	
PACKETLEN	
COLLCONFIG	
TX_BD_NUM	
MIIMODE	
MIICOMMAND	
MIIADDRESS	
MIITX_DATA	
MIIRX_DATA	
MIISTATUS	
MAC_ADDR0	
MAC_ADDR1	

名称	描述
HASH0_ADDR	
HASH1_ADDR	
TXCTRL	

### 24.8.1 MODE

地址: 0x20070000



位	名称	权限	复位值	描述
31:24	RSVD			
23:18	rsvd	rsvd	6'h0	Reserved
17	RMII_EN	r/w	1'b0	RMII mode enable 0: MII PHY I/F is used 1: RMII PHY I/F is used
16	RECSMALL	r/w	1'b0	Receive small frame enable 0: Frames smaller than MINFL are ignored. 1: Frames smaller than MINFL are accepted.
15	PAD	r/w	1'b1	Padding enable 0: Do not add pads to frames shorter than MINFL. 1: Add pads to short frames, until the length equals MINFL.
14	HUGEN	r/w	1'b0	Huge frames enable 0: The maximum frame length is MAXFL. All additional bytes are dropped. 1: Frame size is not limited by MAXFL and can be up to 64K bytes.

位	名称	权限	复位值	描述
13	CRCEN	r/w	1'b1	CRC Enable 0: TX MAC does not append CRC field. 1: TX MAC will append CRC field to every frame.
12:11	rsvd	rsvd	2'b0	Reserved
10	FULLD	r/w	1'b0	Full duplex 0: Half duplex mode. 1: Full duplex mode.
9:7	rsvd	rsvd	3'b0	Reserved
6	IFG	r/w	1'b0	Inter frame gap check 0: IFG is verified before each frame be received. 1: All frames are received regardless to IFG requirement.
5	PRO	r/w	1'b0	Promiscuous mode enable 0: The destination address is checked before receiving. 1: All frames received regardless of the address.
4	rsvd	rsvd	1'b0	Reserved
3	BRO	r/w	1'b1	Broadcast address enable 0: Reject all frames containing the broadcast address unless the PRO bit is asserted. 1: Receive all frames containing broadcast address.
2	NOPRE	r/w	1'b0	No preamble mode 0: 7-byte preamble will be sent. 1: No preamble will be sent.
1	TXEN	r/w	1'b0	Transmit enable 0: Transmitter is disabled. 1: Transmitter is enabled. If TX_BD_NUM equals 0x0 (zero buffer descriptors are used), then the transmitter is disabled regardless of TXEN.

位	名称	权限	复位值	描述
0	RXEN	r/w	1'b0	<p>Receiver enable</p> <p>0: Receiver is disabled.</p> <p>1: Receiver is enabled.</p> <p>If TX_BD_NUM equals 0x80 (all buffer descriptors are used for TX), then the receiver is disabled regardless of RXEN.</p>

#### 24.8.2 INT\_SOURCE

地址: 0x20070004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
	RSVD														

位	名称	权限	复位值	描述
31:7	RSVD			
6	RXC	r/w	1'b0	<p>Receive control frame</p> <p>This bit indicates that the control frame was received. It is cleared by writing 1 to it.</p> <p>Bit RXFLOW in the CTRLMODE register must be set to 1 in order to get the RXC bit set.</p>
5	TXC	r/w	1'b0	<p>Transmit control frame</p> <p>This bit indicates that a control frame was transmitted. It is cleared by writing 1 to it.</p> <p>Bit TXFLOW in the CTRLMODE register must be set to 1 in order to get the TXC bit set.</p>
4	BUSY	r/w	1'b0	<p>Busy</p> <p>This bit indicates that RX packet is being received and there is no empty buffer descriptor to use. It is cleared by writing 1 to it.</p> <p>This bit appears regardless to the IRQ bits in the Receive Buffer Descriptor.</p>

位	名称	权限	复位值	描述
3	RXE	r/w	1'b0	<p>Receive error</p> <p>This bit indicates that an error occurred while receiving data (overrun, receiver error, dribble nibble, too long, &gt;64K, CRC error, bus error or late collision. It is cleared by writing 1 to it.</p> <p>This bit appears only when IRQ bit is set in the Receive Buffer Descriptor.</p>
2	RXB	r/w	1'b0	<p>Receive frame</p> <p>This bit indicates that a frame was received. It is cleared by writing 1 to it.</p> <p>This bit appears only when IRQ bit is set in the Receive Buffer Descriptor.</p>
1	TXE	r/w	1'b0	<p>Transmit error</p> <p>This bit indicates that a buffer was not transmitted due to a transmit error (underrun, retransmission limit, late collision, bus error or defer timeout). It is cleared by writing 1 to it.</p> <p>This bit appears only when IRQ bit is set in the Transmit Buffer Descriptor.</p>
0	TXB	r/w	1'b0	<p>Transmit buffer</p> <p>This bit indicates that a buffer has been transmitted. It is cleared by writing 1 to it.</p> <p>This bit appears only when IRQ bit is set in the Transmit Buffer Descriptor.</p>

### 24.8.3 INT\_MASK

地址: 0x20070008

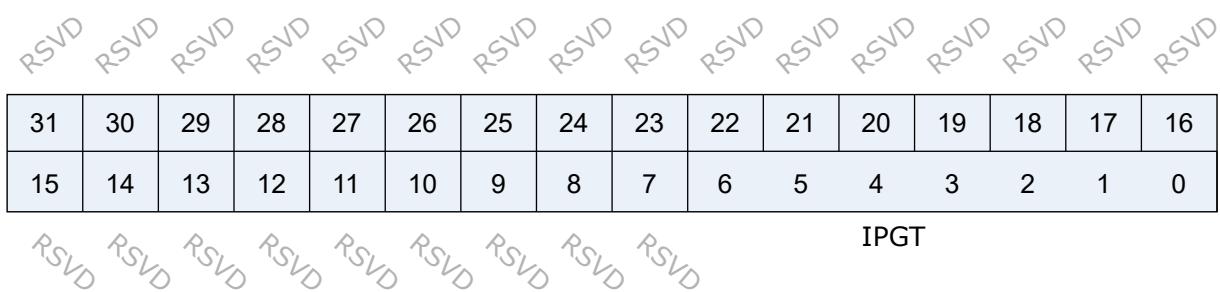
| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |      |      |

位	名称	权限	复位值	描述
31:7	RSVD			

位	名称	权限	复位值	描述
6	RXC_M	r/w	1'b1	Receive control frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
5	TXC_M	r/w	1'b1	Transmit control frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
4	BUSY_M	r/w	1'b1	Busy mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
3	RXE_M	r/w	1'b1	Receive error mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
2	RXB_M	r/w	1'b1	Receive frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
1	TXE_M	r/w	1'b1	Transmit error mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
0	TXB_M	r/w	1'b1	Transmit buffer mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked

#### 24.8.4 IPGT

地址: 0x2007000c



位	名称	权限	复位值	描述
31:7	RSVD			

位	名称	权限	复位值	描述
6:0	IPGT	r/w	7'h18	<p>Inter packet gap</p> <p>The recommended value is 0x18 (24 clock cycles), which equals 9.6 us for 10 Mbps and 0.96 us for 100 Mbps mode</p>

#### 24.8.5 PACKETLEN

地址: 0x20070018

MINFL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MAXFL

位	名称	权限	复位值	描述
31:16	MINFL	r/w	16'h40	<p>Minimum frame length</p> <p>The minimum Ethernet packet is 64 bytes long (0x40).</p> <p>To receive small packets, assert the RECSMALL bit or change the MINFL value.</p> <p>To transmit small packets, assert the PAD bit or change the MINFL value.</p>
15:0	MAXFL	r/w	16'h600	<p>Maximum frame length</p> <p>The maximum Ethernet packet is 1518 bytes long. To support this and to have some additional space for tags, a default maximum packet length equals to 1536 bytes (0x600).</p> <p>For bigger packets, you can assert the HUGEN bit or increase the value of MAXFL field.</p>

## 24.8.6 COLLCONFIG

地址: 0x2007001c

RSVD	MAXRET													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

RSVD COLLVALID

位	名称	权限	复位值	描述
31:20	RSVD			
19:16	MAXRET	r/w	4'hF	<p>Maximum retry</p> <p>This field specifies the maximum number of consequential retransmission attempts after the collision is detected.</p> <p>When the maximum number has been reached, the TX MAC reports an error and stops transmitting the current packet.</p> <p>According to the Ethernet standard, the MAXRET default value is set to 0xf (15).</p>
15:6	RSVD			
5:0	COLLVALID	r/w	6'h3F	<p>Collision valid</p> <p>This field specifies a collision time window. A collision that occurs later than the time window is reported as a "Late Collisions" and transmission of the current packet is aborted.</p>

## 24.8.7 TX\_BD\_NUM

地址: 0x20070020

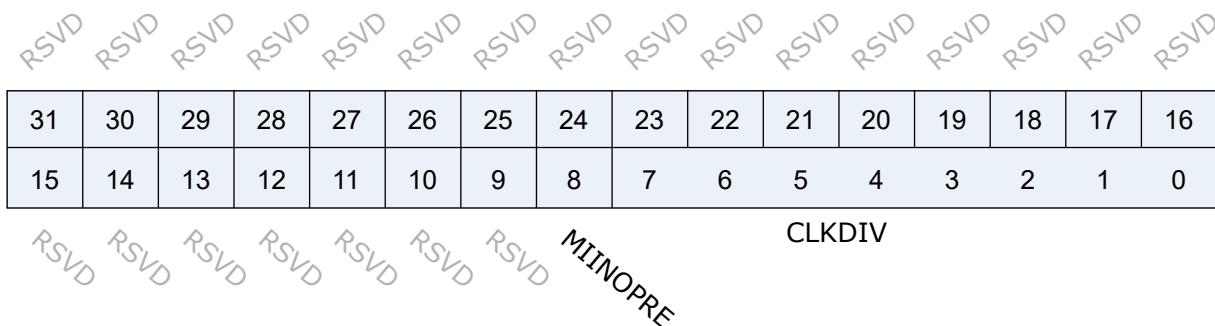
RSVD	RXBDPTR								RSVD	TXBDPTR					
31	30	29	28	27	26	25	24	23		22	21	20	19	18	17
15	14	13	12	11	10	9	8	7		6	5	4	3	2	1

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD TXBDNUM

位	名称	权限	复位值	描述
31	RSVD			
30:24	RXBDPTR	r	7'h0	RX buffer descriptors (BD) pointer, pointing at the RXBD currently being used
23	RSVD			
22:16	TXBDPTR	r	7'h0	TX buffer descriptors (BD) pointer, pointing at the TXBD currently being used
15:8	RSVD			
7:0	TXBDNUM	r/w	8'h40	TX buffer descriptors (BD) number Number of TX BD. TX and RX share 128 (0x80) descriptors, so the number of RX BD equals 0x80 - TXBDNUM. The maximum number of TXBDNUM is 0x80. Values greater than 0x80 cannot be written into this register.

#### 24.8.8 MIIMODE

地址: 0x20070028



位	名称	权限	复位值	描述
31:9	RSVD			
8	MIINOPRE	r/w	1'b0	No preamble for Management Data (MD) 0: 32-bit preamble will be sent. 1: No preamble will be sent.
7:0	CLKDIV	r/w	8'h64	Clock divider for Management Data Clock (MDC) The source clock is bus clock and can be divided by any even number.

## 24.8.9 MIICOMMAND

地址: 0x2007002c

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD WCTRLDATA RSTAT SCANSTAT

位	名称	权限	复位值	描述
31:3	RSVD			
2	WCTRLDATA	r/w	1'b0	Write control data, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time
1	RSTAT	r/w	1'b0	Read status, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time
0	SCANSTAT	r/w	1'b0	Scan status, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time

## 24.8.10 MIIADDRESS

地址: 0x20070030

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD RSVD RSVD RGAD RSVD RSVD FIAD

位	名称	权限	复位值	描述
31:13	RSVD			
12:8	RGAD	r/w	5'h0	Register Address
7:5	RSVD			
4:0	FIAD	r/w	5'h0	PHY Address

#### 24.8.11 MIITX\_DATA

地址: 0x20070034

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

CTRLDATA

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	CTRLDATA	r/w	16'h0	Control Data to be written to PHY

#### 24.8.12 MIIRX\_DATA

地址: 0x20070038

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |

PRSD

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	PRSD	r	16'h0	Received Data from PHY

## 24.8.13 MIISTATUS

地址: 0x2007003c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |      |      |

位	名称	权限	复位值	描述
31:2	RSVD			
1	MIIM_BUSY	r	1'b0	MIIM I/F busy signal 0: The MIIM I/F is ready. 1: The MIIM I/F is busy.
0	MIIM_LINKFAIL	r	1'b0	MIIM I/F link fail signal

## 24.8.14 MAC\_ADDR0

地址: 0x20070040

MAC_B2								MAC_B3							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC_B4								MAC_B5							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

位	名称	权限	复位值	描述
31:24	MAC_B2	r/w	8'd0	Ethernet MAC address byte 2
23:16	MAC_B3	r/w	8'd0	Ethernet MAC address byte 3
15:8	MAC_B4	r/w	8'd0	Ethernet MAC address byte 4
7:0	MAC_B5	r/w	8'd0	Ethernet MAC address byte 5

### 24.8.15 MAC\_ADDR1

地址: 0x20070044

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |      |      |

MAC\_B0

MAC\_B1

位	名称	权限	复位值	描述
31:16	RSVD			
15:8	MAC_B0	r/w	8'd0	Ethernet MAC address byte 0
7:0	MAC_B1	r/w	8'd0	Ethernet MAC address byte 1

### 24.8.16 HASH0\_ADDR

地址: 0x20070048

HASH0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

HASH0

位	名称	权限	复位值	描述
31:0	HASH0	r/w	32'h0	Lower 32-bit of HASH register

### 24.8.17 HASH1\_ADDR

地址: 0x2007004c

HASH1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

HASH1

位	名称	权限	复位值	描述
31:0	HASH1	r/w	32'h0	Upper 32-bit of HASH register

## 24.8.18 TXCTRL

地址: 0x20070050

RSVD	TXPAUSERQ															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

TXPAUSESETV

位	名称	权限	复位值	描述
31:17	RSVD			
16	TXPAUSERQ	r/w	1'b0	TX Pause Request Writing 1 to this bit starts sending control frame and is automatically cleared to zero.
15:0	TXPAUSESETV	r/w	16'h0	TX Pause Timer Value The value that is sent in the pause control frame.

## 25.1 简介

USB(Universal Serial Bus) 通用串行总线，是一个外部总线标准，用于规范电脑与外部设备的连接和通讯。BL808 支持 USB2.0 (HighSpeed + FullSpeed)，可作为主机控制器 (Host) 或者外围控制器 (Device)。作为主机控制器时，它包含一个支持所有速度事务的 USB 主机控制器。在没有软件干预的情况下，主机控制器可以自行处理基于事务的数据结构以减轻 CPU 的负载并自动在 USB 总线上发送和接收数据。当作为外围控制器时，除端点 0 外的每个端点都支持 USB 规范的传输类型，以适配各种应用类型。此外，BL808 的 USB 控制器符合 OTG 标准，支持会话请求协议 (SRP) 和主机协商协议 (HNP)。

## 25.2 主要特征

- 兼容 USB2.0 (HighSpeed + FullSpeed)
- 兼容 OTG revision1.3
- 支持 UTMI+ Level 3
- 支持 OTG SRP 和 HNP 协议
- 支持 Host、OTG、Device 模式
- 支持 LPM
- 兼容 EHCI 数据结构 (不支持 FSTN 和 SITD)
- 支持 9 个端点 (EP0: control endpoint EP1-EP8: interrupt/isolation/bulk endpoint)
- 支持 1 个控制传输专用 FIFO 和 4 个一般 FIFO (控制 FIFO: 64 字节一般 FIFO: 512 字节)
- 支持双 FIFO、三 FIFO 模式工作
- 支持 DMA
- 支持 VDMA

## 25.3 功能描述

### 25.3.1 USB 使用步骤

1. 配置内部 transceiver
2. 配置 usb controller
3. 配置 usb 中断
4. 配置 usb dma/vdma (不支持 cpu 直接读写 fifo)
5. 完成

### 25.3.2 部分寄存器配置及功能描述

### 25.3.3 USB 枚举阶段中断处理流程

## 26.1 简介

ISO11898 是一种国际标准，是目前工业和汽车领域中应用最广泛的现场总线之一。

## 26.2 主要特征

- 支持自定义位速率
- 支持 ISO11898 协议 2.0A 和 2.0B
- 支持自测模式（自发自收）
- 支持帧过滤
- 支持静默模式（不应答，没有有效的错误标志）
- 支持不重发的单次传输
- 支持查询仲裁失败的 bit 位
- 任何总线错误都可以触发中断

## 26.3 功能介绍

### 26.3.1 发送缓冲区（TXB）

发送缓冲区是 CPU 和位流处理器之间的接口，能够存储完整的消息，以便通过 ISO11898 网络传输。缓冲区长度为 13 字节，由 CPU 写入，由位流处理器读取。

### 26.3.2 接收缓冲区（RXB, RXFIFO）

接收缓冲区是接收滤波器和 CPU 之间的接口，用于储存从 ISO11898 总线接收并且经过滤波的消息。接收缓冲区（RXB）表示接收 FIFO（RXFIFO）中可由 CPU 访问的 13 字节窗口，其总长度为 64 字节。接收 FIFO 使得 CPU 可以在处理一帧消息的同时接收其他消息。

### 26.3.3 接收过滤器 (ACF)

接收过滤器将收到的标识符和接收过滤寄存器中的内容进行比较，并决定是否应该接受此消息。如果接受此消息，则将完整的消息储存在 RXFIFO 中。

### 26.3.4 位流处理器 (BSP)

位流处理器是一个序列发生器，用于处理发送缓冲区、接收缓冲区和 ISO11898 总线之间的数据。它还在 ISO11898 总线上执行错误侦测、仲裁、位填充和错误处理。

### 26.3.5 位时序逻辑 (BTL)

位时序逻辑监控串行 ISO11898 总线并处理与总线相关的位时序。它在消息开始时总线由“隐形到显性”的转换时进行位同步（硬同步），也在后续的消息接收期间再次进行位同步（软同步）。BTL 还提供可编程的时间段用来补偿传播延时与相移（例如由于振荡器漂移），并可定义采样点和在一个位时间内的采样次数。

### 26.3.6 错误管理逻辑 (EML)

错误管理逻辑负责传输层模块的错误界定。它从 BSP 接收错误声明然后将错误的统计信息通知给 BSP 和 IML（中断管理逻辑）

## 26.4 功能描述

### 26.4.1 模式

#### 26.4.1.1 自测模式

通过对 MOD 寄存器的 STM 位置'1' 来选择自测模式。在这种模式下，可以使用接收请求命令在总线上无其他活跃节点的情况下进行全节点测试，并且即使没有收到应答 ISO11898 控制器也将执行成功的传输。

#### 26.4.1.2 静默模式

通过对 MOD 寄存器的 LOM 位置'1' 来选择进入静默模式。在这种模式下，ISO11898 控制器即使成功收到消息也不会对 ISO11898 总线做出应答，并且错误计数器也会停留在当前值。这种操作模式会强制 ISO11898 控制器成为被动错误，此时也不可以传输消息。在软件驱动的位速率检测和热插拔等场景中可以使用这种静默模式，其他所有功能都可以和正常模式一样使用。

#### 26.4.1.3 复位模式

MOD 寄存器中的 RM 位一旦由'0' 变成'1'，将导致当前的发送和接收消息都被终止并进入复位模式。当 RM 位从'1' 到'0' 转变时，ISO11898 控制器将返回到操作模式。

在不同模式下的不同操作含义如下表所示。

表 26.1: 不同模式下各寄存器含义说明

ADDRESS OFFSET	OPERATING MODE			RESET MODE	
	READ	WRITE		READ	WRITE
0x00	mode		mode	mode	mode
0x04	(00H)		command	(00H)	command
0x08	status		reserved	status	reserved
0x0C	interrupt		reserved	interrupt	reserved
0x10	interrupt enable		interrupt enable	interrupt enable	interrupt enable
0x14	reserved		reserved	reserved	reserved
0x18	bus timing 0		reserved	bus timing 0	bus timing 0
0x1C	bus timing 1		reserved	bus timing 1	bus timing 1
0x20	reserved		reserved	reserved	reserved
0x24	reserved		reserved	reserved	reserved
0x28	reserved		reserved	reserved	reserved
0x2C	arbitration lost capture		reserved	arbitration lost capture	reserved
0x30	error code capture		reserved	error code capture	reserved
0x34	error warning limit		reserved	error warning limit	error warning limit
0x38	RX error counter		reserved	RX error counter	RX error counter
0x3C	TX error counter		reserved	TX error counter	TX error counter
0x40	SFF RX frame information	EFF RX frame information	SFF TX frame information	EFF TX frame information	acceptance code 0
0x44	RX identifier 1	RX identifier 1	TX identifier 1	TX identifier 1	acceptance code 1
0x48	RX identifier 2	RX identifier 2	TX identifier 2	TX identifier 2	acceptance code 2
0x4C	RX data 1	RX identifier 3	TX data 1	TX identifier 3	acceptance code 3
0x50	RX data 2	RX identifier 4	TX data 2	TX identifier 4	acceptance mask 0
0x54	RX data 3	RX data 1	TX data 3	TX data 1	acceptance mask 1
0x58	RX data 4	RX data 2	TX data 4	TX data 2	acceptance mask 2
0x5C	RX data 5	RX data 3	TX data 5	TX data 3	acceptance mask 3
0x60	RX data 6	RX data 4	TX data 6	TX data 4	reserved
0x64	RX data 7	RX data 5	TX data 7	TX data 5	reserved
0x68	RX data 8	RX data 6	TX data 8	TX data 6	reserved

表 26.1: 不同模式下各寄存器含义说明 (continued)

ADDRESS OFFSET	OPERATING MODE				RESET MODE	
	READ		WRITE		READ	WRITE
0x6C	(FIFO RAM)	RX data 7	reserved	TX data 7	reserved	reserved
0x70	(FIFO RAM)	RX data 8	reserved	TX data 8	reserved	reserved
0x74	RX message counter		reserved		RX message counter	reserved
0x78	RX buffer start address		reserved		RX buffer start address	RX buffer start address
0x7C	clock divider		clock divider		clock divider	clock divider

## 26.4.2 发送处理

### 26.4.2.1 发送流程

1. 检查 SR 寄存器的 TBS 位来确保发送缓冲区是空的。
2. 配置帧信息、ID 号和数据。
3. 通过置位 CMR 寄存器中的 TR 位来请求发送。

### 26.4.2.2 终止发送

当 CPU 要求暂停先前的发送时，可以使用终止发送的功能，例如需要先发送更紧急的消息。已经在发送过程中的消息不受此功能影响不会停止。为了查看之前的消息是否发送成功，应该检查 SR 寄存器中的发送完成标志位 (TCS)。应用软件可以通过对 CMR 寄存器中的 AT 位置'1' 来使用该功能，这应该在 SR 寄存器中的 TBS 位为'1' 或者发送中断产生后执行。

有一点需要注意的是，即使消息被终止，也会产生发送中断，因为发送缓冲区的状态位已经指示为“已释放”状态。

### 26.4.2.3 自发自收

应用软件可以通过置位 CMR 寄存器中的 SRR 位实现自发自收，此时发送和接收是同步进行的。其他操作与普通发送流程一样。

### 26.4.2.4 注意点

1. 如果同时置位 CMA 寄存器中的 TR 和 AT 位，消息将会只发送一次。此时即使出现错误事件或者仲裁失败也不会再次发送。
2. 如果同时置位 CMA 寄存器中的 SRR 和 AT 位，消息将使用自发自收的方式只发送一次。此时即使出现错误事件或者仲裁失败也不会再次发送。
3. 如果同时置位 CMA 寄存器中的 SRR、TR 和 AT 位，消息将以同时置位 TR 和 AT 位的方式发送。
4. 一旦状态寄存器中的发送状态位被置位，内部的发送请求位就被自动清零。
5. 如果 CMA 寄存器中的 TR 和 SRR 被同时置位，SRR 位将被忽略。

### 26.4.3 接收处理

#### 26.4.3.1 接收流程

接收到的消息被储存在 64 字节深度的内部 FIFO 中，FIFO 完全由硬件管理，从而节省了 CPU 的处理负荷，简化了软件并保证了数据的一致性。应用程序可以通过 FIFO 的输出接口来读取收到的消息。当 SR 寄存器中的 RBS 置位时，RXFIFO 中则有一帧或多帧消息可读，软件获取消息后，将 CMR 寄存器中的 RRB 置位可释放当前消息占用的 RXFIFO。

#### 26.4.3.2 消息数量

RMC 寄存器表示 RXFIFO 中的可读消息的数量，该值随每一次的接收事件递增，并随每一次的释放缓冲区递减。复位后该值是 0。

#### 26.4.3.3 接收缓冲区

RBSA 寄存器表示当前内部 RAM 中储存的接收到的消息的第一个字节的地址，其映射到接收缓冲区窗口。借助该信息可以解读内部 RAM 中的内容。这部分的内部 RAM 区域可以由 CPU 进行读取和写入（仅在复位模式下可写入）。

示例：如果 RBSA 的值为 18H，则接收缓冲区窗口（偏移地址为 10H 到 12H）的当前可读消息也被存储在 RAM 地址从 18H 开始的位置。由于 RAM 地址是直接映射到 ISO11898 偏移地址 20H（对应 RAM 地址 0H）开始的位置，所以消息也可以从 ISO11898 偏移地址 38H 和后面的字节中读出。（ISO11898 地址 = RBSA + 20H = 18H + 20H = 38H）。如果消息地址超过 RAM 地址 3FH 则它将从 RAM 地址 0 继续。

当 FIFO 中至少还有一条消息时，应该发出释放接收缓冲区的命令，此时 RBSA 就被更新到下一条消息的开始位置。

在硬件复位时，RBSA 寄存器的值被初始化为 ‘00H’，在软件复位（设置为复位模式）时该寄存器值不会变化，但 FIFO 被清除，这意味着 RAM 内容不会改变，但是下一个接收（或发送）的消息将覆盖接收缓冲区窗口中的可见消息。

## 26.4.4 标识符过滤

在接收过滤器的协助下，只有当接收到的消息的标识符位与接收过滤器寄存器中的预定义位相同时，ISO11898 控制器才能允许将接收到的消息传递到 RXFIFO。接收过滤器由接收码寄存器（ACRn）和接收屏蔽寄存器（AMRn）组成。可被接收的消息中的匹配位由接收码寄存器设定，哪些位可以屏蔽由接收屏蔽寄存器设定。

有两种不同的滤波模式可以选择（MOD 寄存器中的 AFM 位设定）：

- 单滤波器模式 (AFM = 1).
- 双滤波器模式 (AFM = 0).

### 26.4.4.1 单滤波器配置

在这种配置中，可以定义一个长达 4 字节的滤波器。过滤字节和消息字节之间的位对应关系取决于当前接收的帧格式。

**标准帧：**如果接收到标准帧格式的消息，包括 RTR 位和前两个数据字节在内的完整标识符用于接受过滤。如果由于设置了 RTR 位而没有数据字节存在，或者由于设定了相应数据长度而没有数据字节或只有一个数据字节，也可以接收到消息。

所有滤波位之间是逻辑与的关系，必须所有位都通过滤波器，一条消息才能被接收到。请注意 AMR1 和 ACR1 的低 4 位没有使用，为了与将来产品兼容这几位应该被设置为屏蔽位，即 AMR1 的 3~0 位都为 1。

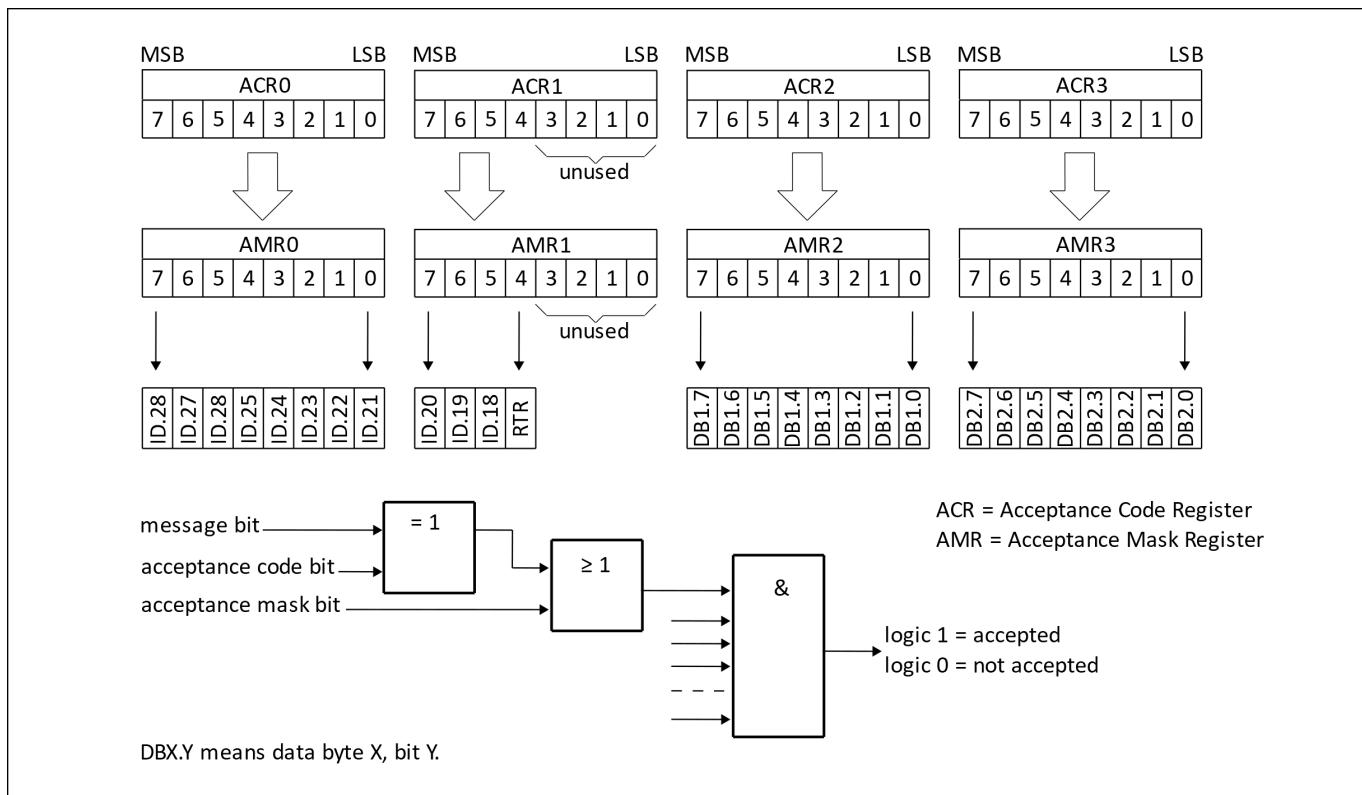


图 26.1: Single filter configuration, receiving standard frame messages

**扩展帧：**如果接收到扩展帧格式的消息，包括 RTR 位在内的完整标识符用于接受过滤。

所有滤波位之间是逻辑与的关系，必须所有位都通过滤波器，一条消息才能被接收到。请注意 AMR3 和 ACR3 的低 2 位没有使用，

为了与将来产品兼容这几位应该被设置为屏蔽位，即 AMR3 的 1~0 位都为 1。

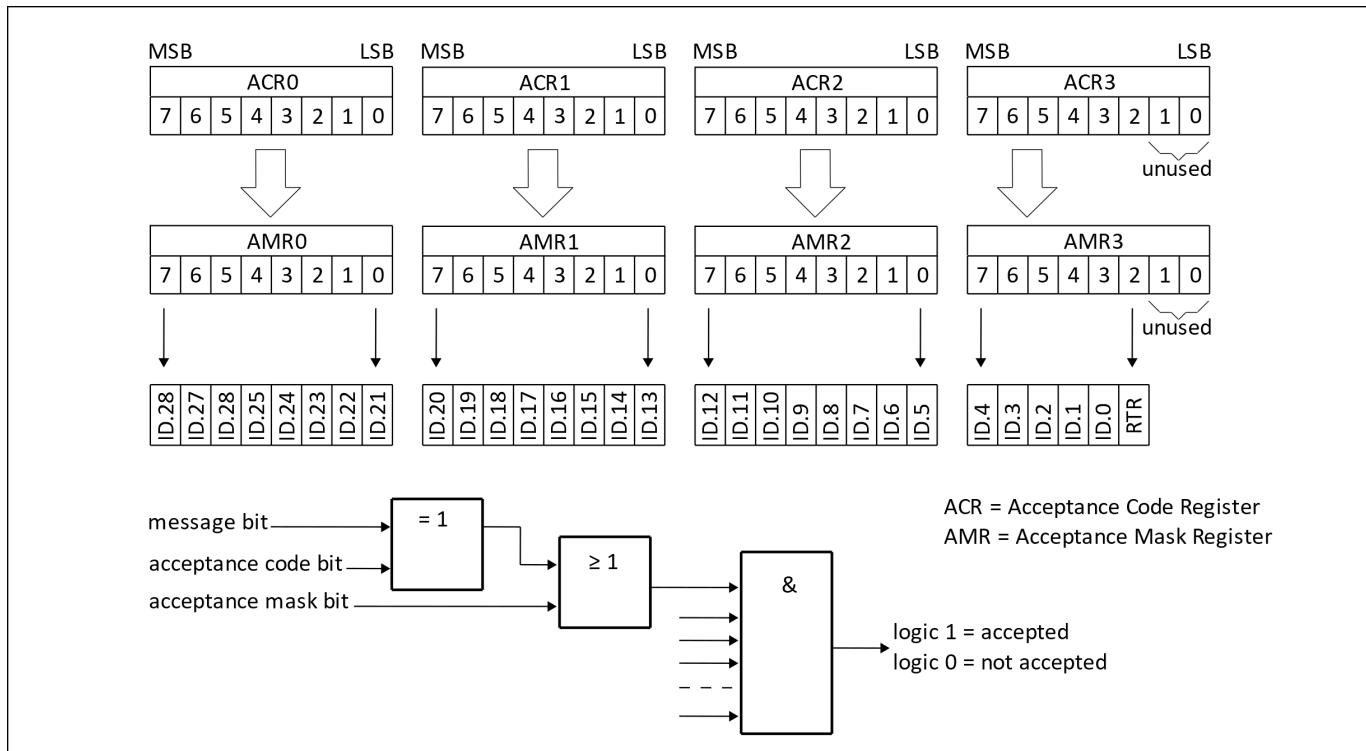


图 26.2: Single filter configuration, receiving extended frame messages

#### 26.4.4.2 双滤波器配置

可以在这种配置中定义两个短滤波器，收到的消息会与这两个滤波器都进行比较，以决定是否将该消息复制到接收缓冲区。只要有一个滤波器接收该消息，则收到的消息就是有效的。过滤字节和消息字节之间的位对应关系取决于当前接收的帧格式。

标准帧：如果接收到标准帧格式的消息，则定义的这两个滤波器看起来有点不一样，第一个滤波器比较包括 RTR 和第一个数据字节在内的完整的标识符，第二个滤波器则只比较包括 RTR 位在内的标准标识符。

为了成功接收消息，至少一个完整过滤器的所有单个位比较都表示接受。在 RTR 被置位或者数据长度为 0 的情况下是没有数据的。然而，如果直到 RTR 位之前的第一部分都表示接受，则消息也是可以通过滤波器 1 的。

如果第一个滤波器不需要过滤数据字节，则 AMR1 和 AMR3 的低 4 位必须设置为逻辑 1（无关紧要），然后这两个滤波器使用包括 RTR 在内的标准标识符一样地运行。

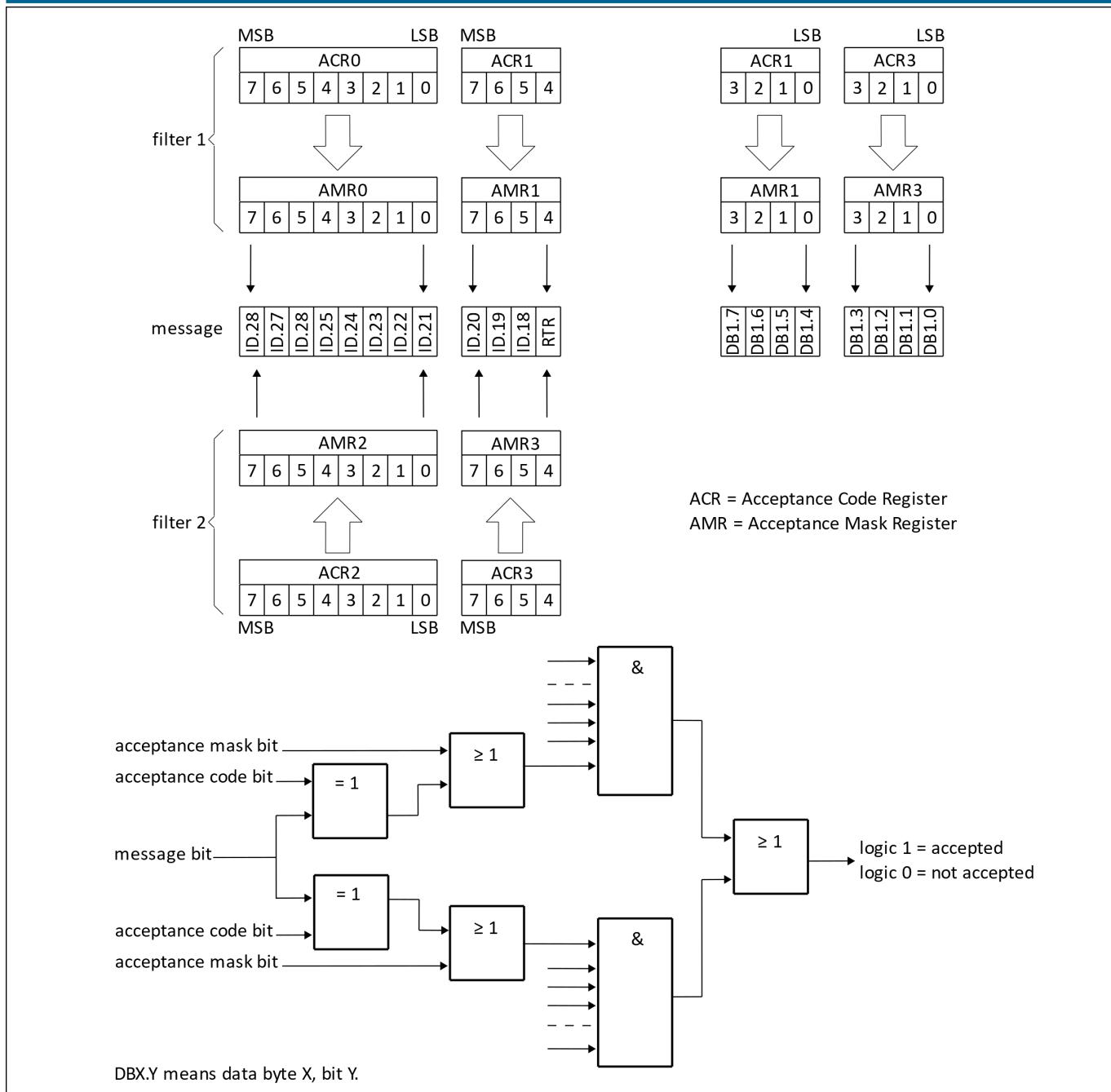


图 26.3: Dual filter configuration, receiving standard frame messages

**扩展帧:** 如果接收到扩展帧格式的消息，则定义的这两个滤波器看起来是一样的。这两个滤波器都只比较扩展标识符的前两个字节。

至少一个完整的滤波器的所有单个位比较都表明接受，消息才能被成功接收。

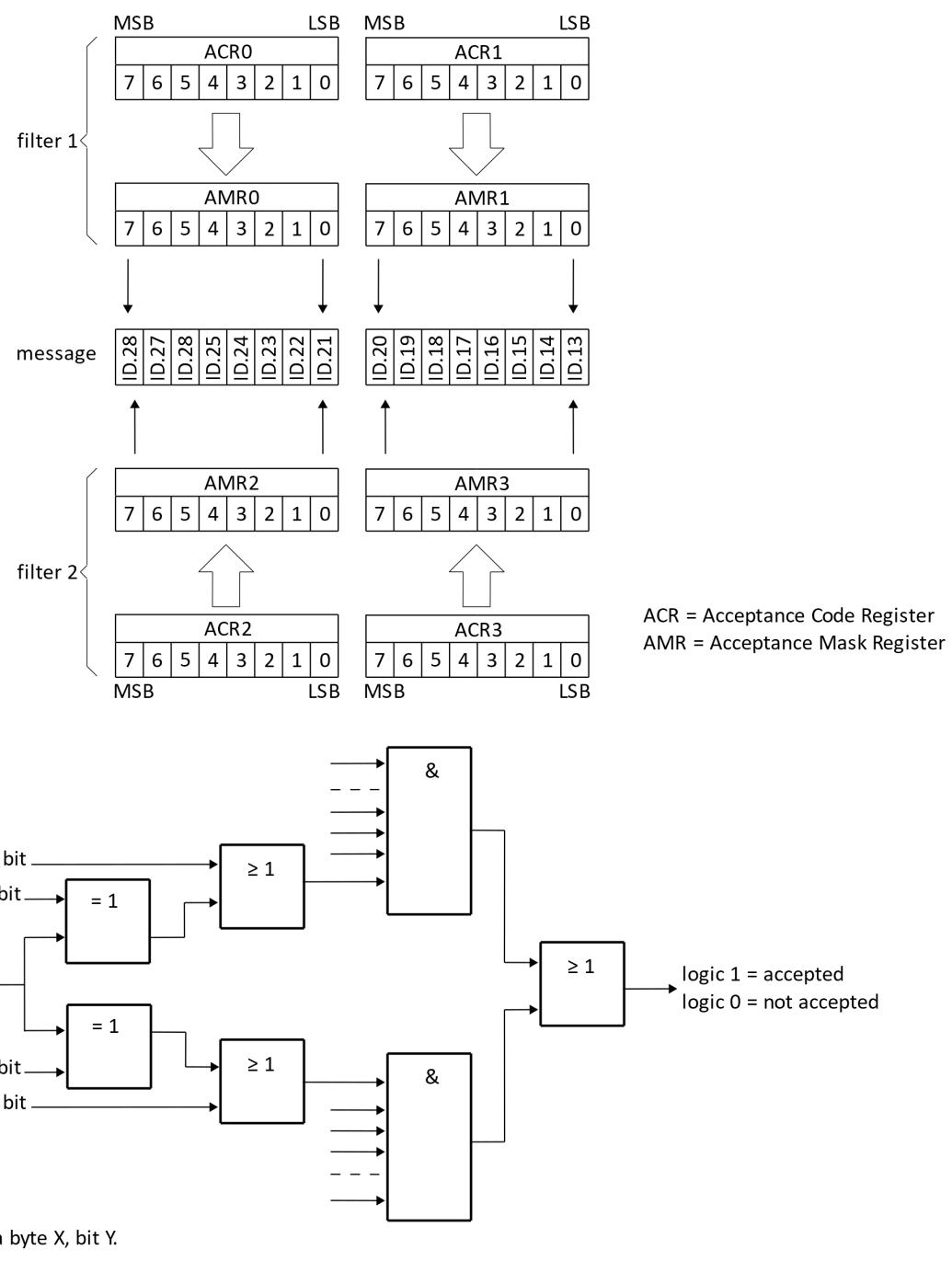


图 26.4: Dual filter configuration, receiving extended frame messages

## 26.4.5 出错管理

### 26.4.5.1 仲裁失败

仲裁失败捕获寄存器（ALC）包含仲裁失败的位置，该寄存器只能被 CPU 读取不能写入。如果使能了仲裁失败的中断，则一旦仲裁失败将产生中断。同时，位流处理器中的当前位的位置将被捕获到 ALC 中。在用户软件读取 ALC 的内容之前，该寄存器的值将一直固定不变。读取该寄存器值后，捕获机制就会再次激活。在中断寄存器被读出的时候，相应的中断标志也是被清除的，在中断失败寄存器被读出之前是不会再次产生仲裁失败中断的。

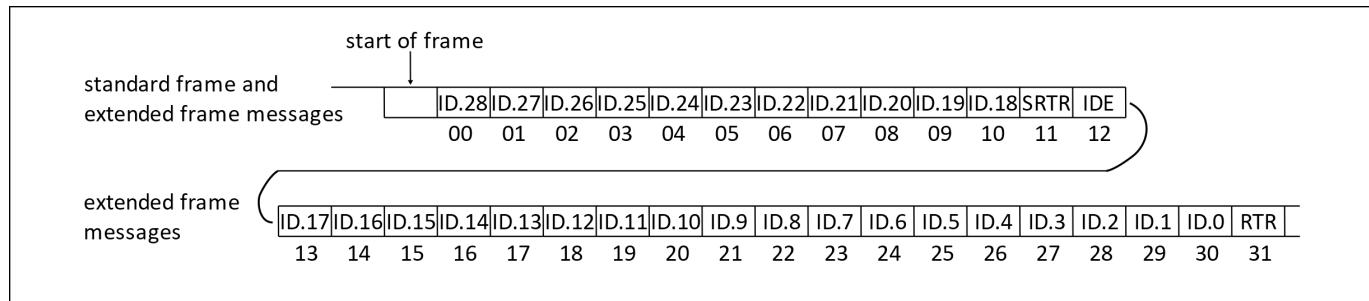


图 26.5: Arbitration lost bit number interpretation

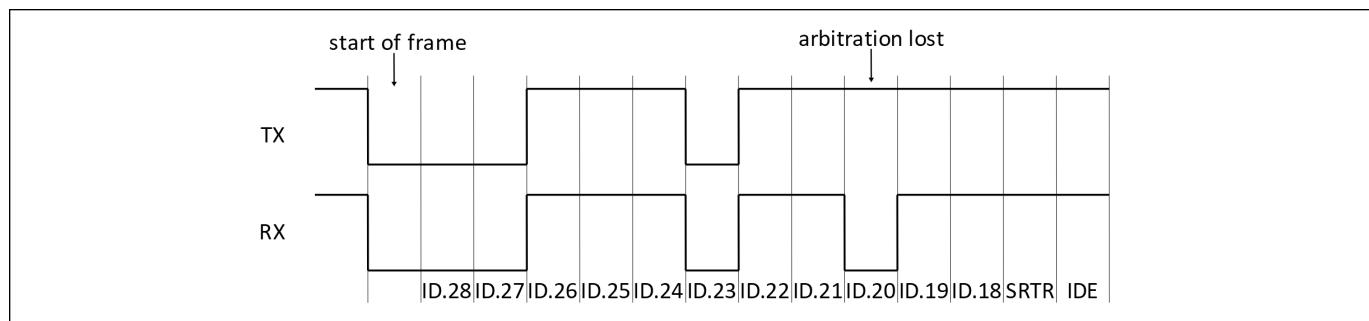


图 26.6: Example of arbitration lost bit number interpretation; result: ALC = 08

表 26.2: 仲裁失败捕获的位置

BITS					DECIMAL VALUE	FUNCTION
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	00	arbitration lost in bit 1 of identifier
0	0	0	0	1	01	arbitration lost in bit 2 of identifier
0	0	0	1	0	02	arbitration lost in bit 3 of identifier
0	0	0	1	1	03	arbitration lost in bit 4 of identifier
0	0	1	0	0	04	arbitration lost in bit 5 of identifier
0	0	1	0	1	05	arbitration lost in bit 6 of identifier
0	0	1	1	0	06	arbitration lost in bit 7 of identifier
0	0	1	1	1	07	arbitration lost in bit 8 of identifier
0	1	0	0	0	08	arbitration lost in bit 9 of identifier

表 26.2: 仲裁失败捕获的位置 (continued)

BITS					DECIMAL VALUE	FUNCTION
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	1	0	0	1	09	arbitration lost in bit 10 of identifier
0	1	0	1	0	10	arbitration lost in bit 11 of identifier
0	1	0	1	1	11	arbitration lost in bit SRTR
0	1	1	0	0	12	arbitration lost in bit IDE
0	1	1	0	1	13	arbitration lost in bit 12 of identifier
0	1	1	1	0	14	arbitration lost in bit 13 of identifier
0	1	1	1	1	15	arbitration lost in bit 14 of identifier
1	0	0	0	0	16	arbitration lost in bit 15 of identifier
1	0	0	0	1	17	arbitration lost in bit 16 of identifier
1	0	0	1	0	18	arbitration lost in bit 17 of identifier
1	0	0	1	1	19	arbitration lost in bit 18 of identifier
1	0	1	0	0	20	arbitration lost in bit 19 of identifier
1	0	1	0	1	21	arbitration lost in bit 20 of identifier
1	0	1	1	0	22	arbitration lost in bit 21 of identifier
1	0	1	1	1	23	arbitration lost in bit 22 of identifier
1	1	0	0	0	24	arbitration lost in bit 23 of identifier
1	1	0	0	1	25	arbitration lost in bit 24 of identifier
1	1	0	1	0	26	arbitration lost in bit 25 of identifier
1	1	0	1	1	27	arbitration lost in bit 26 of identifier
0	1	1	0	0	28	arbitration lost in bit 27 of identifier
1	1	1	0	1	29	arbitration lost in bit 28 of identifier
1	1	1	1	0	30	arbitration lost in bit 29 of identifier
1	1	1	1	1	31	arbitration lost in bit RTR

### 26.4.5.2 错误捕获

错误捕获寄存器 (ECC) 包含总线错误的类型和位置，该寄存器只能被 CPU 读取不能写入。如果使能了总线错误中断，则一旦总线发生错误将产生总线错误中断。同时，位流处理器中的当前位的位置将被捕获到 ECC 中。在用户软件读取 ECC 的内容之前，该寄存器的值将一直固定不变。读取该寄存器值后，捕获机制就会再次激活。对中断寄存器中相应位的读取操作会清除该位，在读取捕获寄存器之前，不会再次产生总线错误中断。

错误捕获寄存器中值代表的错误类型和种类如下所示。

表 26.3: 错误捕获的类型

BIT ECC.7	BIT ECC.6	FUNCTION
0	0	bit error
0	1	form error
1	0	stuff error
1	1	other type of error

表 26.4: 错误捕获的位置

BIT ECC.4	BIT ECC.3	BIT ECC.2	BIT ECC.1	BIT ECC.0	FUNCTION
0	0	0	1	1	start of frame
0	0	0	1	0	ID.28 to ID.21
0	0	1	1	0	ID.20 to ID.18
0	0	1	0	0	bit SRTR
0	0	1	0	1	bit IDE
0	0	1	1	1	ID.17 to ID.13
0	1	1	1	1	ID.12 to ID.5
0	1	1	1	0	ID.4 to ID.0
0	1	1	0	0	bit RTR
0	1	1	0	1	reserved bit 1
0	1	0	0	1	reserved bit 0
0	1	0	1	1	data length code
0	1	0	1	0	data field
0	1	0	0	0	CRC sequence
1	1	0	0	0	CRC delimiter
1	1	0	0	1	acknowledge slot
1	1	0	1	1	acknowledge delimiter
1	1	0	1	0	end of frame
1	0	0	1	0	intermission
1	0	0	0	1	active error flag
1	0	1	1	0	passive error flag
1	0	0	1	1	tolerate dominant bits
1	0	1	1	1	error delimiter
1	1	1	0	0	overload flag

### 26.4.5.3 接收错误计数器 (RXERR)

接收错误计数寄存器的值代表当前接收错误的数量，在硬件复位后该寄存器被初始化为逻辑 0。在操作模式下该寄存器只能被 CPU 执行读取操作，对该寄存器的写操作只能在复位模式下执行。如果发生总线关闭事件，RXERR 被设置为逻辑 0。此时，总线处于关闭状态，对该寄存器的写操作不起作用。

需要注意的是，只有先进入复位模式，CPU 才可以对 RXERR 的值进行修改，在这样的情况下，错误状态可能会改变，错误警告中断和错误被动中断不会发生，除非再取消复位模式。

### 26.4.5.4 发送错误计数器 (TXERR)

发送错误计数寄存器的值代表当前发送错误的数量，在操作模式下该寄存器只能被 CPU 执行读取操作，对该寄存器的写操作只能在复位模式下执行。在硬件复位后该寄存器的值被初始化为逻辑 0。如果发生总线关闭事件，TXERR 值就被设定为 127，这样就可以计算协议定义的最短时间（出现 128 次总线空闲信号）。在此期间读取该寄存器的值可以获取总线关闭恢复的状态信息。如果总线处于关闭状态，则对 TXERR 的范围从 0 到 254 的写操作会清除总线关闭状态标志，并且控制器将在清除复位模式后等待 11 个连续隐形位（总线空闲）出现一次。

通过 CPU 将 255 写入 TXERR 将产生总线关闭事件，需要注意的是只有先进入复位模式才可以进行 CPU 强制修改该寄存器值的操作，这样的情况下，错误状态或者总线状态将可能改变，错误警告中断或错误被动中断不会受新值影响，除非再退出复位模式。退出复位模式后，TXERR 的值还好像和发生总线错误导致总线关闭那样一样的机制运行，这意味着会再次进入复位模式，TXERR 的值又被初始化为 127，RXERR 的值被初始化为 0，并且相关状态和中断寄存器都被重新设置。此时退出复位模式将执行协议定义的总线关闭恢复流程（等待 128 个总线空闲信号的发生）。如果在总线关闭并恢复前 ( $\text{TXERR} > 0$ ) 再次进入复位模式，总线将继续保持关闭状态并且 TXERR 的值被冻结。

### 26.4.5.5 错误限值设定

错误警告限制可以由 EWLR 寄存器设定，该寄存器默认值（硬件复位后）是 96。在复位模式下，这个寄存器可被 CPU 读取或写入，在操作模式下，该寄存器只能被读取。当 RXERR 和 TXERR 两个错误计数值至少一个大于等于 EWLR 寄存器设定的值时，SR 寄存器中的 ES 位将被置位，否则被清零，此时如果 IER 寄存器中的 EIE 位被置位，则将产生错误警告中断。需要注意的是该寄存器只有在先进入复位模式后才能操作。对该寄存器的操作可能会引起错误状态的改变，并且不会让错误警告中断产生，除非再退出复位模式。

## 26.4.6 位时序

时序图如下：

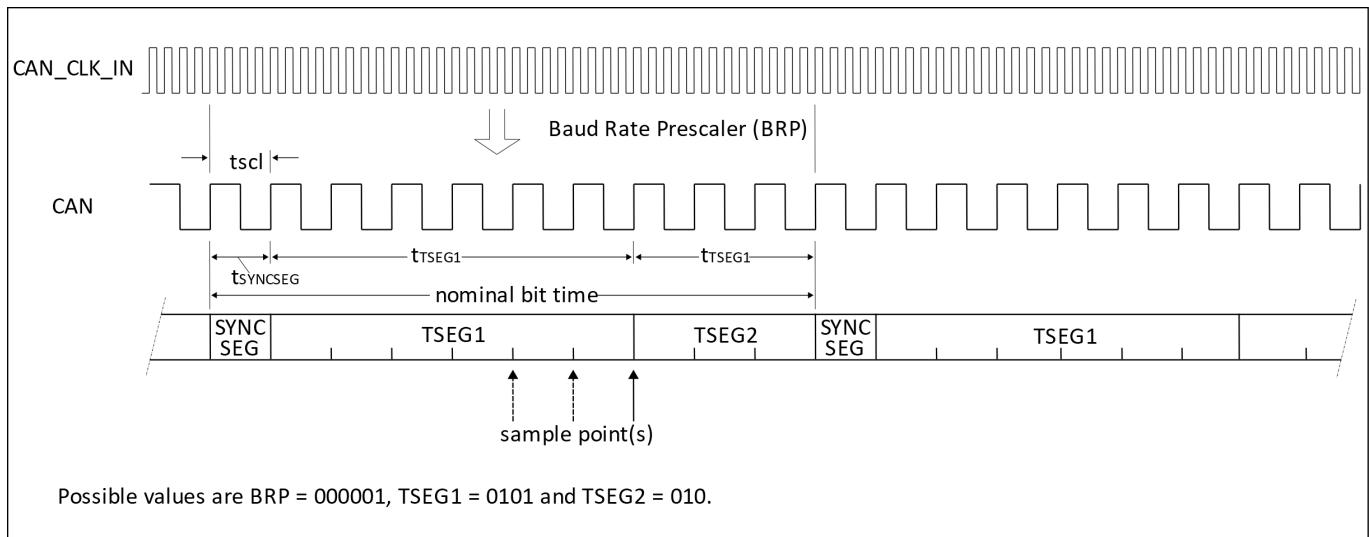


图 26.7: General structure of a bit period

### 26.4.6.1 波特率分频器 (BRP)

ISO11898 控制器的系统时钟  $tscl$  的周期是可以设定的，并且这确定了各个位的时序。ISO11898 系统时钟的计算公式如下：

$$tscl = 2 * tCLK * (32 * BRP.5 + 16 * BRP.4 + 8 * BRP.3 + 4 * BRP.2 + 2 * BRP.1 + BRP.0 + 1)$$

### 26.4.6.2 同步跳转宽度 (SJW)

为了补偿不同总线控制器的时钟振荡器之间的相移，任何总线控制器都必须在当前传输的任何相关信号边缘重新同步。同步跳转宽度定义了一个位周期可以通过一次重新同步缩短或延长的最大时钟周期数：

$$tSJW = tscl * (2 * SJW.1 + SJW.0 + 1)$$

### 26.4.6.3 采样 (SAM)

当 BTR1 寄存器中的 SAM 位为 1 时，总线将采样三次，这种模式推荐在中低速总线中使用，此时总线中的滤波器将有好处。如果 SAM 位为 0，则总线只采样一次，这种模式推荐在高速模式中使用。

### 26.4.6.4 时间段 (TSEG)

TSEG 包含 BTR1 寄存器中 TSEG1 和 TSEG2 两部分，它决定了每一个位的时钟数和采样点位置，计算公式如下：

$$t_{SYNCSEG} = 1 * tscl$$

$$t_{TSEG1} = tscl * (8 * TSEG1.3 + 4 * TSEG1.2 + 2 * TSEG1.1 + TSEG1.0 + 1)$$

$$t_{TSEG2} = tscl * (4 * TSEG2.2 + 2 * TSEG2.1 + TSEG2.0 + 1)$$

## 27.1 简介

MJPEG(Motion Joint Photographic Experts Group) 是一种视频编码格式，可精确到帧编辑和多层图像处理，把运动的视频序列作为连续的静止图像来处理，这种压缩方式单独完整地压缩每一帧。通过对 YCbCr 格式的原始数据进行压缩，可以大幅降低一帧图像所占用的内存空间。

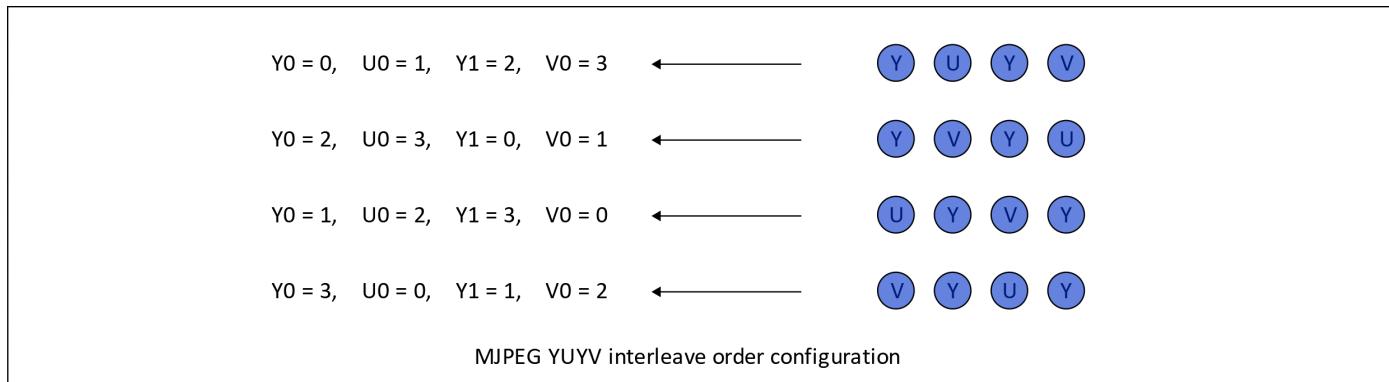
## 27.2 主要特征

- 可配置的输入格式，包括：
  - YCbCr4:2:2 平面或打包模式
  - YCbCr4:2:0 平面模式
  - YCbCr4:0:0
- 可配置的输入数据 Y、Cb、Cr 排列顺序
- 量化系数表可自由配置
- 支持软件模式和连动模式
- 支持 swap 模式
- 支持 kick 模式
- 可预留 jpg 头部空间和自动添加 jpg 尾
- 连续缓存多达 4 组图片信息
- 多种应用中断，有利于弹性使用与出错提示

## 27.3 功能描述

### 27.3.1 输入配置

通过寄存器 MJPEG\_CONTROL\_1 的位 <REG\_YUV\_MODE> 可以选择输入数据的 YCbCr 格式，包括 YCbCr4:2:2 平面或打包模式，YCbCr4:2:0 平面模式和 YCbCr4:0:0 灰度图。当选择 YCbCr4:2:2 打包模式时，通过寄存器 MJPEG\_HEADER\_BYTE 的高 8 位可以配置 Y、Cb、Cr 的排列顺序。在选择其他模式时，可以通过寄存器 MJPEG\_CONTROL\_1 的位 <REG\_ORDER\_U\_EVEN> 设定 Cb 和 Cr 的顺序。详细的配置如下图所示。



### 27.3.2 量化系数表

量化系数表可由用户自由配置，<reg\_q\_0\_00> 到 <reg\_q\_0\_3F> 表示灰度信息 Y 分量的量化表，<reg\_q\_1\_00> 到 <reg\_q\_1\_3F> 表示色度信息 Cb 和 Cr 的量化表。量化表按照从左上角向下排列的顺序，第一列结束后紧跟第二列，即 reg\_q\_0\_00 代表 Y 分量第一行第一列的量化数值，reg\_q\_0\_01 代表 Y 分量第二行第一列的量化数值，reg\_q\_0\_07 代表 Y 分量第八行第一列的量化数值，reg\_q\_0\_08 代表 Y 分量第一行第二列的量化数值，依次类推。

### 27.3.3 软件模式和连动模式

将 MJPEG\_CONTROL\_2 寄存器的位 <REG\_MJPEG\_SW\_MODE> 置 1 时即使能软件模式，在此模式下 MJPEG 会从内存中读取指定帧数的原始图片数据进行压缩，此模式下需要预先将图片数据准备好。

将 MJPEG\_CONTROL\_2 寄存器的位 <REG\_MJPEG\_SW\_MODE> 清 0 时即使能连动模式，在此模式下 MJPEG 会将 CAM 模块的输出作为其输入进行处理，MJPEG 是以 8\*8 的数据块为一个单元进行处理的，当 CAM 向内存中写完 8 行数据后，MJPEG 就会开始工作，MJPEG 处理完的内存空间会释放给 CAM 重新使用，这样 CAM 无需一整张图片的内存空间即可完成与 MJPEG 的连动。

### 27.3.4 swap 模式

使用 swap 模式时，MJPEG 存储空间会被平均分成两块，当 MJPEG 写完其中一块时会产生一个中断通知软件将数据读走，而它会将数据写入另一块中。来回交替使用，从而用不足一帧图片的存储空间进行数据处理。

### 27.3.5 kick 模式

将 MJPEG\_CONTROL\_2 寄存器的位 <REG\_SW\_KICK\_MODE> 置 1 时即使能 kick 模式,在此模式下每次向 MJPEG\_CONTROL\_2 寄存器的位 <REG\_SW\_KICK> 写 1 即可启动一次压缩, 压缩行数由 MJPEG\_YUV\_MEM\_SW 寄存器的位 <REG\_SW\_KICK\_HBLK> 确定。需要注意的是 kick 模式只能在软件模式下使用, 不能在连动模式下使用。

### 27.3.6 jpg 功能

jpg 功能会自动在每帧数据的开头留出一定字节数的空间并填零, 该空间的大小由寄存器 MJPEG\_HEADER\_BYTE 的低 12 位进行设置。另外如果使能自动填充 jpg 尾的话还会在每帧数据的结尾补上两个字节数据, 这两个字节的值为 0xFFD9。

### 27.3.7 缓存图片信息

模块内部包含 4 组 FIFO 记录图片地址、图片大小和量化系数。每当此模块完整写入一帧到内存, 便会将此帧图片的起始地址、图片大小和量化系数纪录于此 FIFO 中, 但要注意的是当发生内存剩余不足, 或是 4 组 FIFO 满存的状况时, 模块会自动丢掉接下来图片的讯息, 在图片信息取出的部分, 可通过 APB 接口做 pop 的动作, 将最旧的图片信息空出, 此时 FIFO 会自动推进, 保证 FIFO 内部图片信息的时序。

### 27.3.8 支持多种中断信息 (可独立开关配置)

- A、Normal 中断-可设定固定写入几张图片后发出中断
- B、Camera 中断-MJPEG 处理的速度跟不上 CAM 模块写入的速度导致 CAM 溢出时, 发出中断
- C、Memory 中断-当内存被复写时, 发出中断
- D、Frame 中断-当未处理图片超过 4 组时, 发出中断
- E、Swap 中断 - 当 swap 模式下一个内存块被写满时, 发出中断

## 28.1 简介

JDEC 可将内存中的 jpg 压缩图片解码成原始图片，进一步通过 YUV2RGB 模块转换成 RGB 模式进行显示。

## 28.2 主要特点

- 支持 YUV400/YUV420/YUV422 格式
- 量化系数可调，范围是 1~75、100
- 支持 swap 模式
- 支持 skip 模式，自动跳过 jpg 头
- 有多达 16 张 jpg 图片的缓冲区
- 支持灵活的中断配置模式，既可设定指定张数中断，也可以设置全部完成中断

## 28.3 功能描述

### 28.3.1 输出配置

通过寄存器 JDEC\_CONTROL\_1 的位 <REG\_YUV\_MODE> 可以选择输出数据的 YCbCr 格式，包括 YCbCr4:2:2 平面模式，YCbCr4:2:0 平面模式和 YCbCr4:0:0 灰度图。输出图像数据的起始地址由寄存器 JDEC\_YY\_FRAME\_ADDR 和 JDEC\_UV\_FRAME\_ADDR 决定。输出图像数据的分辨率由寄存器 JDEC\_FRAME\_SIZE 决定，该寄存器中位 <REG\_FRAME\_HBLK> 代表垂直方向 8x8 块的数量，位 <REG\_FRAME\_WBLK> 代表水平方向 8x8 块的数量。

### 28.3.2 量化系数

通过寄存器 JDEC\_CONTROL\_1 的位 <REG\_Q\_MODE> 可以设置量化系数，当设置值不超过 75 时，量化系数为设置值；当设置值超过 75 时，量化系数为 100。

### 28.3.3 swap 模式

使用 swap 模式时，会有两帧输出图像大小的空间作为缓冲区，当其中一帧空间放满后则会继续将解压后的图像数据放在另一块帧空间缓冲区里，来回交替使用。该模式可以和显示模块连动提高效率。

### 28.3.4 skip 模式

当将寄存器 JDEC\_HEADER\_SKIP 中位 <REG\_HDER\_SKIP> 设置为 1 时，则 skip 模式被启用，此时 JDEC 模块会以寄存器 JDEC\_FRAM\_PUSH 寄存器中位 <REG\_JP\_ADDR> 设定的 jpg 图片起始地址加上寄存器 JDEC\_HEADER\_SKIP 中位 <REG\_HDER\_SKIP\_BYTE> 设定的偏移字节数量作为最终解压 jpg 的实际起始地址。这种操作方式可以用在跳过 jpg 头部信息的场景中。

### 28.3.5 缓冲区

JDEC 输入缓冲区可以存放多达 16 张 jpg 的相关信息，每次向 JDEC\_FRAM\_PUSH 寄存器的位 <REG\_JP\_PUSH> 写 1 都会使缓冲区可用空间递减，JDEC 每次解压完一张 JPG 后该缓冲区的可用空间会递增，可通过读取 JDEC\_FRAM\_STS 寄存器中的位 <JP\_FRAME\_CNT> 获取当前缓冲区中待解压的 jpg 图片数量。JDEC\_CONTROL\_3 寄存器中的位 <FRAME\_VALID\_CNT> 表示当前解压完成但没有 POP 的图片数量。

### 28.3.6 操作流程

- 失能 JDEC 模块，将寄存器 JDEC\_CONTROL\_1 中位 REG\_MJ\_DEC\_ENABLE 清零。
- 设置输出图像数据的模式为 YUV422/YUV420/YUV400 等。
- 根据应用场景选择是否使能 swap 模式。
- 设置图像的量化系数。
- 设定输出图像 Y 和 UV 分量分别在内存中的起始地址。
- 根据数据图像数据的分辨率设置水平和垂直的 8x8 块数量。
- 如果需要使能 skip 模式，则需要设置 skip 的字节数量。
- 使能 JDEC 模块，将寄存器 JDEC\_CONTROL\_1 中位 REG\_MJ\_DEC\_ENABLE 置 1。
- 同步写入输入 jpg 图像的起始地址和加入缓冲区的命令，即将 jpg 图像的起始地址写入寄存器 JDEC\_FRAM\_PUSH 且该寄存器的位 <REG\_JP\_PUSH> 为 1。

## 28.4 使用注意事项

- JPG 图片在内存中的起始地址需要 8 字节对齐
- JPG 图片的分辨率必须是 8 的整数倍
- 目前仅支持标准的霍夫曼编码

## 28.5 寄存器描述

## 29.1 简介

VENC 采用 H264 视频编码标准, 主要是以预测及运动补偿等方式进行压缩, 并以环路滤波提升画质, 兼顾码流传输和图像品质要求

## 29.2 主要特点

- 1920x1080p@30fps + 640x480@30fps, BP/MP
- 输入: Semi-Planar YCbCr 4:2:0
- 输出: NALU(Network Abstract Layer UInt) in byte stream format
- CBR/VBR mode
- 最大 8 个 ROI
- 最大 16 个 OSD 编码区域
- 支持软件模式和连动模式
- 可动态配置最大/最小量化参数
- 可动态配置 I/P 帧目标位元
- 可动态配置 I 帧距离

## 29.3 功能描述

### 29.3.1 软件模式和连动模式

软件模式: 硬件以帧为单位和软件进行交互, 软件设定一个帧启动後硬件才压缩一帧, 完成後回传帧中断。输入帧内存需至少一帧。  
连动模式: 软件设定 H264\_ENCODER\_CTRL 的位 <CFG\_S\_ENC\_SEQ\_EN> 或 <CFG\_ENC\_SEQ\_EN> 来使能序列 (sequence) 编码, 硬件自行侦测输入帧状态并进行压缩, 直到软件拉低序列使能, 硬件才停止压缩 (结束於完整的一帧) 且回传序列中断。序列中每帧做完也会回传帧中断。

### 29.3.2 CBR/VBR 模式

CORE\_REG23 的 <NUM\_IMB\_BITS> 或 <S\_NUM\_IMB\_BITS> 设 0, 则该码流为 VBR 模式, 非 0 则为 CBR 模式; 在 VBR 模式需设定 I/P 帧固定量化参数 (CORE\_REG3/CORE\_REG5), 不建议在硬件工作时切换模式。

### 29.3.3 双码流

两个码流的帧使能, 帧中断, 序列使能, 序列中断是独立配置, 双流的帧率基本上需要一致, 但分辨率和其他压缩参数可以不同。

### 29.3.4 ROI(Region of Interest)

单一码流最多可支援 8 个 ROI 区域, 每个区域有各自的使能和起始/结束宏块位置设定, 需将 H264\_ROI\_MODE 的位 <CFG\_ROI\_UPD> 或 H264\_S\_ROI\_MODE 的位 <CFG\_S\_ROI\_UPD> 设 1 使新设定生效; 硬件则会每帧同步 ROI 设定。

### 29.3.5 OSD(On-Screen Display) 编码区域

硬件最多支持 16 个 OSD 区域, 软件设定 H264 OSD\_EN 来决定每个码流 OSD 区域, OSD 区域是以起始/结束宏块位置来表示。

### 29.3.6 动态配置最大/最小量化参数

为提供更好的调控码率, 软件设定 CORE\_REG31 之后, 需设 H264\_ENCODER\_CTRL(位 <CFG\_QR\_UPD> 或 <CFG\_S\_QR\_UPD> 分别设定两个码流) 使新设定值生效, 硬件会以 GOP 为单位来同步新设定值, 达成动态调整量化参数的范围。

### 29.3.7 动态配置 I/P 帧目标位元

软件可以设定 CORE\_REG23/CORE\_REG24 来调整 I/P 帧的目标位元; CORE\_REG23/CORE\_REG24 设定后需要再设 H264\_ENCODER\_CTRL(位 <CFG\_QR\_UPD> 或 <CFG\_S\_QR\_UPD>) 分别设定两个码流) 让新的设定值生效, 硬件会以 GOP 为单位来同步新的设定值 (CBR 模式)。

### 29.3.8 动态配置 I 帧距离

软件设定 CORE\_REG8 之后再设定 H264\_ENCODER\_CTRL(位 <CFG\_QR\_UPD> 或 <CFG\_S\_QR\_UPD> 分别设定两个码流) 让新的设定值生效, 硬件会以 GOP 为单位来同步新的设定值 (CBR/VBR 可用)。

### 29.3.9 视频序列中断

在连动模式时使用, 当软件拉低序列使能后, 硬件处理当下该帧后发出序列中断 (VDO\_INT 的位 <S\_SEQ\_DONE\_INT> 或 <SEQ\_DONE\_INT>), 此中断需要软件设定中断清除 (VDO\_INT\_CLR 的位 <S\_SEQ\_DONE\_INT\_CLR> 或 <SEQ\_DONE\_INT\_CLR>); 此中断可以使用遮挡 (VDO\_INT\_MASK 的位 <S\_SEQ\_DONE\_INT\_MASK> 或 <SEQ\_DONE\_INT\_MASK>) 来忽略。

### 29.3.10 帧中断

在软件模式和连动模式下只要硬件完成一帧压缩就会发出帧中断 (VDO\_INT 的位 <S\_FRM\_DONE\_INT> 或 <FRM\_DONE\_INT>), 此中断需要软件设定中断清除 (VDO\_INT\_CLR 的位 <S\_FRM\_DONE\_INT\_CLR> 或 <FRM\_DONE\_INT\_CLR>); 此中断可以使用遮挡 (VDO\_INT\_MASK 的位 <S\_FRM\_DONE\_INT\_MASK> 或 <FRM\_DONE\_INT\_MASK>) 来忽略。

### 29.3.11 输入缓存溢出状态警示

如果发生视频压缩速度远低於 Camera pixel 输出时会发生缓存溢出情形, 软件可透过 VDO\_SRC\_R\_DBG(位 <SRC\_WR\_OV\_RD>) 或 VDO\_S\_SRC\_R\_DBG(位 <S\_SRC\_WR\_OV\_RD>) 的状态寄存器得知。

## 30.1 图形界面流程图

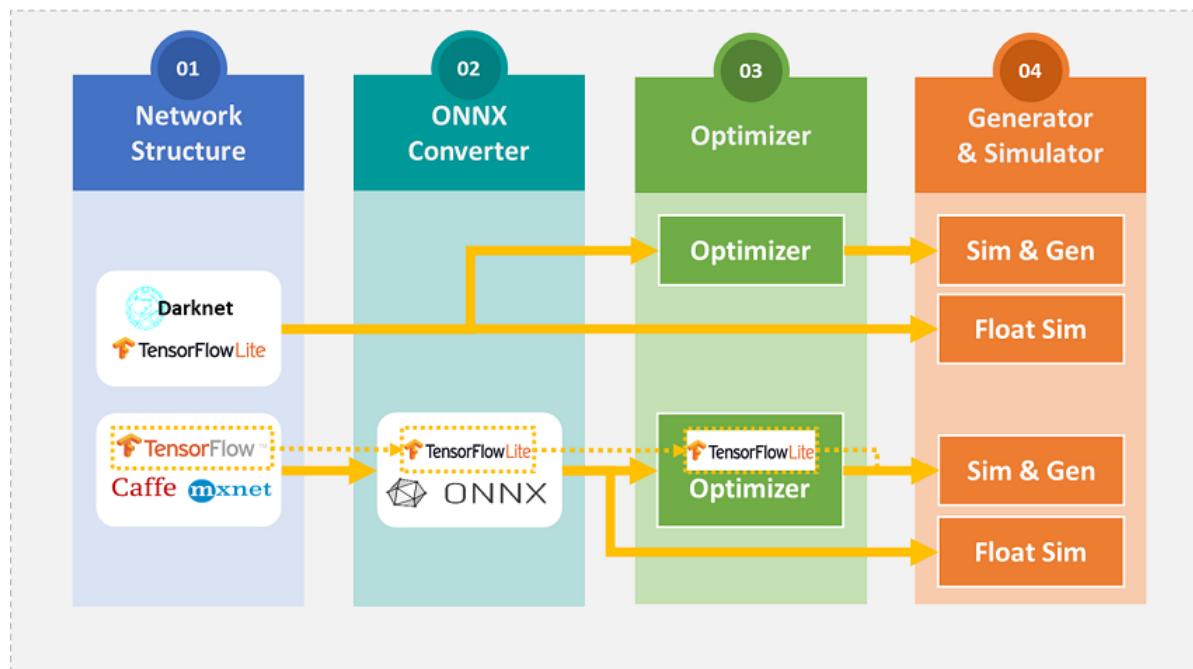


图 30.1: 图形界面流程图

## 30.2 网络结构

要在 GUI 上通过 Netron 显示模型结构，用户需要选择用于显示的平台和配置文件，可选择的平台有：Darknet, Tensorflow, Caffe, Mxnet, ONNX。

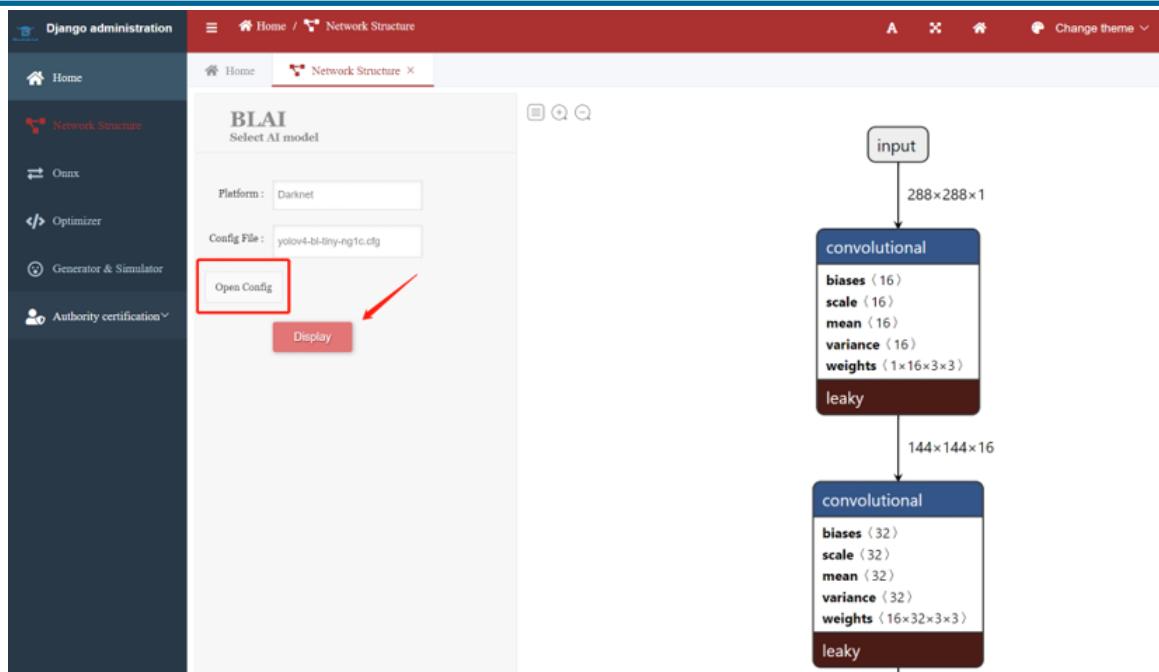


图 30.2: 网络结构图

### 30.3 ONNX 转换器

用户想要将自己的平台转换为 ONNX，必须要选择一个平台并命名 ONNX 文件。可选的平台有：Tensorflow, Caffe, Mxnet (DarkNet、Tensorflow Lite 可直接支持)。

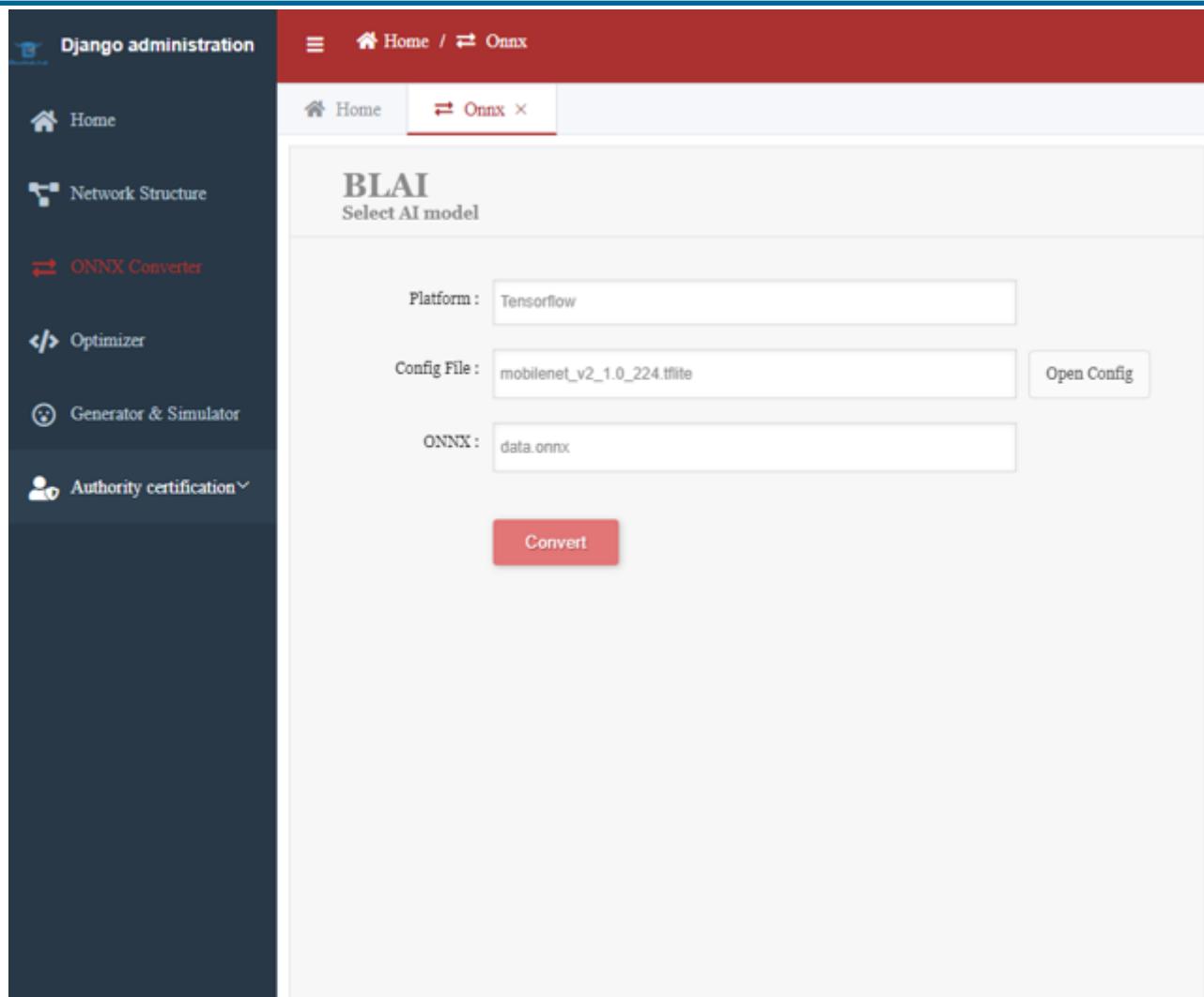


图 30.3: 转换界面

在 ONNX 转换器上按“convert”后，屏幕将显示另外两个按钮，用户可以选择“Display”或“Download”。

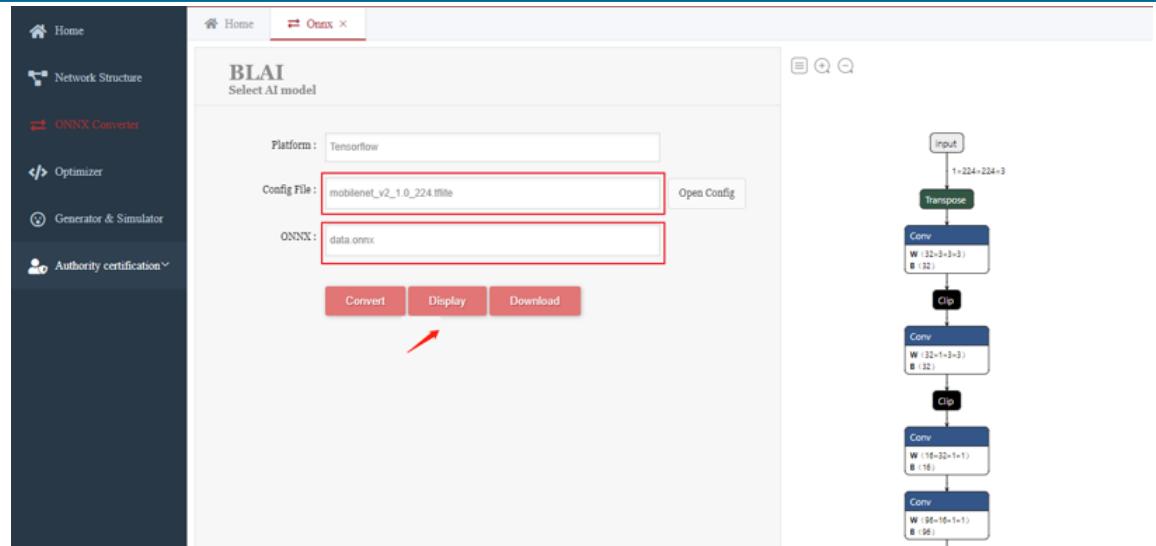


图 30.4: 显示或下载界面

## 30.4 优化器

优化器部分可以将浮点数量化为固定点。

“Optimize”：花更多时间，准确率高

“Fast Optimize”：花费较少的时间，准确性损失值较大

支持的平台有: Darknet, Tensorflow, ONNX (BLAI NPU 也支持 Tensorflow Lite 上的优化器格式)。

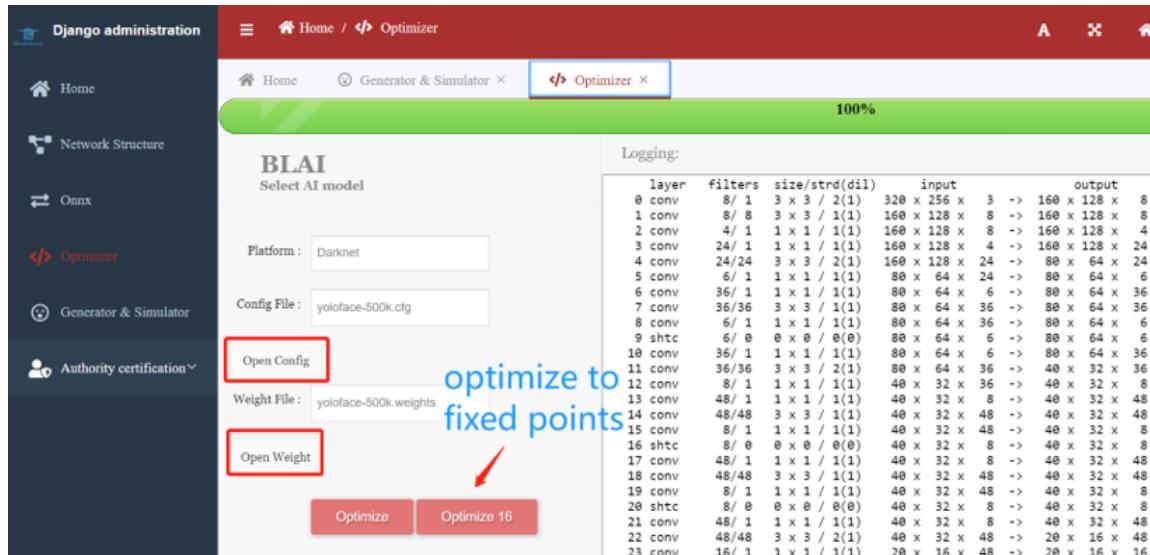


图 30.5: 优化器界面

## 30.5 Generator & Simulator

在做完优化器部分后，点击“Sim&Gen”，使用 8 位量化权重预测图像，并生成 BLAI NPU 指令。

点击“Float Sim”将预测具有 32 位未量化权重的图像。

支持的平台有：Darknet, Tensorflow, ONNX (BLAI NPU 也支持 Tensorflow Lite 上的优化器格式)。

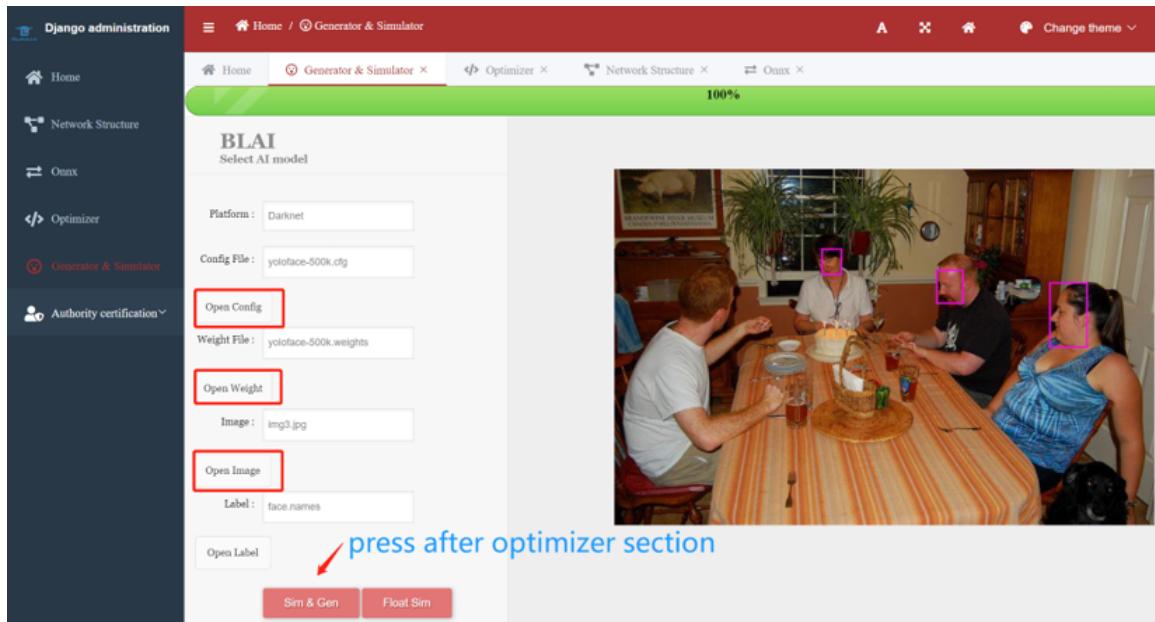


图 30.6: 预测选择界面

# 31

NPU

## 31.1 简介

神经处理单元 (NPU) 或简称为 AI 加速器是一种专用电路，可实现执行机器学习算法所需的所有必要控制和算术逻辑，是为深度学习算法设计的电子电路，通常具有单独的数据存储器和专用指令集架构。NPU 的目标是为深度学习算法提供比一般中央处理单元 (CPU) 更高的效率和性能。NPU 使用大量计算组件来利用高数据级并行性，使用相对较大的缓冲区/内存来利用数据重用模式，以及用于深度学习容错的有限数据宽度运算符。

## 31.2 主要特点

- AI 算力 **0.1 TOPS**
- 支持 8-bit 运算
- 兼容 TensorFlow-lite/ONNX/Caffe/Mxnet/Darknet/Pytorch 模型，可支持多类框架
- 提供 PC 端方便易用的开发工具，包含模型转换、模型量化、性能预估、精度验证
- 支持单层指令模式、多层指令模式
- 支持最大特征图  $4096 \times 4096 \times 4096$ (宽、高、深度)

## 31.3 功能列表

Type	Operators	Applicable Subset Spec.	Processor
Convolution	Conv	1x1 up to 7x7	<b>NPU</b>
	Depthwise Conv	1x1 up to 7x7	<b>NPU</b>
	Pad	same	<b>NPU</b>
	Deconvolution	use Upsample + Conv	<b>NPU</b>
Pooling	MaxPool(2x2)	Stride 2	<b>NPU</b>
	MaxPool(3x3)	3x3	<b>NPU</b>
	AveragePool	Stride 1, 2	DSP
	GlobalAveragePool		DSP
	GlobalMaxPool		DSP

Type	Operators	Applicable Subset Spec.	Processor
Activation	Relu		<b>NPU</b>
	LeakyRelu		<b>NPU</b>
	Relu-n	$n > 0$ (Relu6)	<b>NPU</b>
	Mish	Look up table	<b>NPU</b>
	ELU	Look up table	<b>NPU</b>
	PReLU		DSP
	Sigmoid		DSP
Other processing	BatchNormalization		DSP
	Add (shortcut)		<b>NPU</b>
	Concat (route)		<b>NPU</b>
	Fully Connected		<b>NPU</b>
	Mul		<b>NPU</b>
	Slice		DSP

## 31.4 API 参考

```
void gen_npu_inst_layer(npu_layer* l, bool use_tflite, bool unsgn_input, bool img_in)
```

```
/**
 * function    产生NPU指令
 * @param[in]   1: NPU层数据结构指标
 * @param[in]   use_tflite: 是否使用tflite数据格式
 * @param[in]   unsgn_input: 输入数据是否为uint8
 * @param[in]   img_in: 输入数据是否为YUV400
 * @return      空
 **/
```

```
bool check_BLAI_NPU_RUN(int type, int size, int stride, int dilation)
```

```
/**
 * function    确认该层是否符合NPU加速条件
 * @param[in]   type: 该层类别 (enum LAYER_TYPE)
 * @param[in]   size: 卷积核大小
 * @param[in]   stride: 步伐大小
 * @param[in]   dilation: 空洞卷积核大小
 * @return      true: 符合NPU加速条件, false: 未符合NPU加速条件
 **/
```

```
bool BLAI_MEM_alloc(npu_layer l, PSRAM_ctrl *ctrl)
```

```
/**  
 * function    自动产生NPU相关记忆体配置  
 * @param[in]   l: NPU层数据结构指标  
 * @param[in]   ctrl: 记忆体配置参数结构  
 * @return      true: 记忆体配置成功, false: 记忆体配置失败  
 **/
```

```
void fetch_BLAI_data_general(npu_layer l, PSRAM_ctrl ctrl, bool use_tflite, bool img_in)
```

```
/**  
 * function    自动产生NPU指令 (输入层数量最多为2)  
 * @param[in]   l: NPU层数据结构指标  
 * @param[in]   ctrl: 记忆体配置参数结构  
 * @param[in]   use_tflite: 是否使用tflite数据格式  
 * @param[in]   img_in: 输入数据是否为YUV400  
 * @return      空  
 **/
```

```
void fetch_BLAI_data_route(npu_layer l, PSRAM_ctrl ctrl, bool use_tflite, bool img_in)
```

```
/**  
 * function    针对输入层数量大于两层的ROUTE层, 自动产生NPU指令  
 * @param[in]   l: NPU层数据结构指标  
 * @param[in]   ctrl: 记忆体配置参数结构  
 * @param[in]   use_tflite: 是否使用tflite数据格式  
 * @param[in]   img_in: 输入数据是否为YUV400  
 * @return      空  
 **/
```

```
bool BLAI_encode(npu_layer l, PSRAM_ctrl ctrl, int use_tflite, bool img_in)
```

```
/**  
 * function    自动产生NPU指令、NPU相关记忆体配置  
 * @param[in]   l: NPU层数据结构指标  
 * @param[in]   ctrl: 记忆体配置参数结构  
 * @param[in]   use_tflite: 是否使用tflite数据格式  
 * @param[in]   img_in: 输入数据是否为YUV400  
 * @return      true: 符合NPU加速条件且记忆体配置成功, false: 未符合NPU加速条件或记忆体配置失败  
 **/
```

```
void Load_NPU_weights(npu_layer l, int8_t* WEI_buf, int* BIAS_buf, bool use_tflite)
```

```
/**  
 * function    将该层卷积参数照NPU指定方式存储入NPU专用参数记忆体
```

(continues on next page)

(continued from previous page)

```
* @param[in] l: NPU层数据结构指标
* @param[in] WEI_buf: NPU专用参数记忆体
* @param[in] BIAS_buf: NPU专用参数记忆体
* @param[in] use_tflite: 是否使用tflite数据格式
* @return 空
**/
```

```
void Store_tensor_data_to_NPU(npu_layer l, fixed_point_t* DATA_buf)
```

```
/**
* function 将运算图中的张量数据存储入NPU专用数据记忆体
* @param[in] l: NPU层数据结构指标
* @param[in] DATA_buf: NPU专用数据记忆体
* @return 空
**/
```

```
void Load_NPU_data_to_tensor(npu_layer l, fixed_point_t* DATA_buf)
```

```
/**
* function 从NPU专用数据记忆体读取张量数据
* @param[in] l: NPU层数据结构指标
* @param[in] DATA_buf: NPU专用数据记忆体
* @return 空
**/
```

```
void forward_NPU(npu_layer l, int8_t* DATA_buf, bool use_tflite)
```

```
/**
* function 使用指令运行NPU (需先使用gen_npu_inst_layer产生指令)
* @param[in] l: NPU层数据结构指标
* @param[in] DATA_buf: NPU专用数据记忆体
* @param[in] use_tflite: 是否使用tflite数据格式
* @return 空
**/
```

## 31.5 数据结构参考

**npu\_layer** 数据结构:

```
/** NPU layer information of dynamic fixed point format(CMSIS/NMSIS) */

struct npu_layer {
```

(continues on next page)

(continued from previous page)

```
/////////// Begin of user define region ///////  
  
/** operation type (enum LAYER_TYPE)*/  
uint8_t type;  
  
/** activation type (enum ACTIVATION)*/  
uint8_t activation;  
  
/** layer width */  
uint16_t w;  
  
/** layer height */  
uint16_t h;  
  
/** layer channel (1) */  
uint16_t c;  
  
/** extra layer channel (2-8) */  
uint16_t cn[7];  
  
/** layer output width */  
uint16_t out_w;  
  
/** layer output height */  
uint16_t out_h;  
  
/** layer output channel */  
uint16_t out_c;  
  
/** number of input layers*/  
uint8_t input_num;  
  
/** CONV kernel size */  
uint8_t size;  
  
/** CONV groups size */  
uint16_t groups;  
  
/** CONV dilation size */  
uint8_t dilation;
```

(continues on next page)

(continued from previous page)

```
/** stride size */
uint8_t stride;

/** input tensor type*/
uint8_t input_type;

/** True: combined layer need to keep output data */
bool mid_out;

/** True: input image is 1-channel */
bool img_in;

/** dynamic fixed point format(CMSIS/NMSIS) for input data */
int8_t fdata;

/** dynamic fixed point format(CMSIS/NMSIS) for weight */
int8_t fweight;

/** dynamic fixed point format(CMSIS/NMSIS) for bias */
int8_t fbias;

/** dynamic fixed point format(CMSIS/NMSIS) for output data */
int8_t fout;

/** dynamic fixed point format(CMSIS/NMSIS) for route input data (1) */
int8_t froute1;

/** dynamic fixed point format(CMSIS/NMSIS) for route input data (2) */
int8_t froute2;

/** dynamic fixed point format(CMSIS/NMSIS) for route input data (3-8) */
int8_t frouten[6];

/** Tensorflow-Lite input offset (1) */
uint8_t tf_input1_offset;

/** Tensorflow-Lite input offset (2) */
uint8_t tf_input2_offset;

/** Tensorflow-Lite input offset (3-8) */
uint8_t tf_input_offset_extra[6];

/** Tensorflow-Lite output offset */
```

(continues on next page)

(continued from previous page)

```
uint8_t tf_output_offset;

/** Tensorflow-Lite input shift (1) */
int8_t tf_input1_shift;

/** Tensorflow-Lite input shift (2) */
int8_t tf_input2_shift;

/** Tensorflow-Lite input shift (3-8) */
int8_t tf_input_shift_extra[6];

/** Tensorflow-Lite output shift */
int8_t tf_output_shift;

/** Tensorflow-Lite quantized_activation_min */
int16_t quantized_activation_min;

/** Tensorflow-Lite quantized_activation_max */
int16_t quantized_activation_max;

/** Tensorflow-Lite input multiplier (1) */
int tf_input1_multiplier;

/** Tensorflow-Lite input multiplier (2) */
int tf_input2_multiplier;

/** Tensorflow-Lite input multiplier (3-8) */
int tf_input_multiplier_extra[6];

/** Tensorflow-Lite output multiplier */
int tf_output_multiplier;

/** Tensorflow-Lite route input multiplier */
int tf_route_input_multiplier;

/** Tensorflow-Lite route input multiplier */
int tf_route_input_shift;

/** Tensorflow-Lite left shift */
int8_t tf_left_shift;

/** pointer of input data buffers */
int8_t* input_i8[8];
```

(continues on next page)

(continued from previous page)

```
/** pointer of output data buffer */
int8_t* output_i8;

/** pointer of combined layer output data buffer */
int8_t* mid_output_i8;

/** pointer of weight buffer */
int* weights;

/** pointer of bias buffer */
int* biases;

///////////////////////////////
////// End of user define region //////
///////////////////////////////

/** pointer of NPU instruction buffer */
uint8_t* NPU_inst;

/** flag of NPU processor */
bool NPU_on;

/** memory patch size*/
uint32_t DRAM_patch_size;

/** memory patch location for input data */
uint16_t DRAM_in[8];

/** memory patch location for output data */
uint16_t DRAM_out[8];

/** memory patch location for mid output data */
uint16_t DRAM_mid_out;

/** memory patch location for weight data */
uint16_t DRAM_weight;

/** size of weight data */
int DRAM_nweight;

/** memory patch location for bias data */
uint16_t DRAM_bias;
```

(continues on next page)

(continued from previous page)

```
/** size of bias data */
int DRAM_nbias;

/** uint8_t input data */
bool unsgn_input;

/** number of instruction */
uint8_t inst_cnt;
};
```

## 32.1 简介

IPC(inter-processor communication) 是多核处理器之间通信的一种机制。BL808 有三个异构内核，任意两个内核之间都可以互相通过 IPC 实现通信。

## 32.2 主要特征

- 每个内核都有独立的 32 位 IPC 通道
- 任意两核之间都可以互相通信

## 32.3 功能描述

### 32.3.1 基本原理

每个内核都有一组 IPC 的寄存器，包括 `IPCx_TRI`、`IPCx_STS`、`IPCx_ACK`、`IPCx_IEN`、`IPCx_IDIS`、`IPCxISTS` 共 6 个寄存器，这些寄存器的长度都是 32bits，每个 bit 都对应 IPC 的一个通道。核 M0、LP、D0 分别对应 `IPC0`、`IPC1`、`IPC2`。当一个核需要向另一个核发通知时，只需要向接收核的 `IPCx_TRI` 的对应通道写 1 即可，此时接收核的 `IPCx_STS` 的对应通道即被设置为 1，如果接收核的 IPC 对应通道的中断也被使能，则会收到一个中断，此时即获知了其他核发来的通知。

### 32.3.2 操作流程

以核 x 向核 y 发通知，所使用的通道为 z，其操作流程如下：

- 核 y 使能通道 z 的中断，即核 y 执行操作 `IPCy_IEN = 1<<z`
- 核 x 向 `IPCy_TRI` 寄存器的第 z 位写入 1，即核 x 执行操作 `IPCy_TRI = 1<<z`
- 核 y 收到中断，向核 x 做出应答，即核 y 执行操作 `IPCy_ACK = 1<<z`
- 核 y 执行通知的应用层相关操作

工作流程图如下所示：

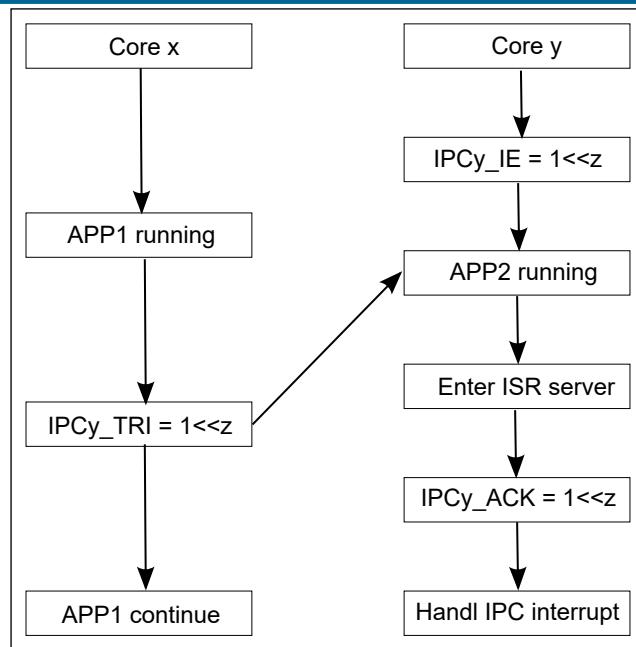


图 32.1: IPC 流程图

### 33.1 概述

低功耗是物联网应用的一项重要指标。芯片的处理器包含 3 种功耗模式，包含工作模式、空闲省电模式和休眠模式，可以根据当前应用场景选择合适的功耗模式，降低芯片功耗延长电池寿命。

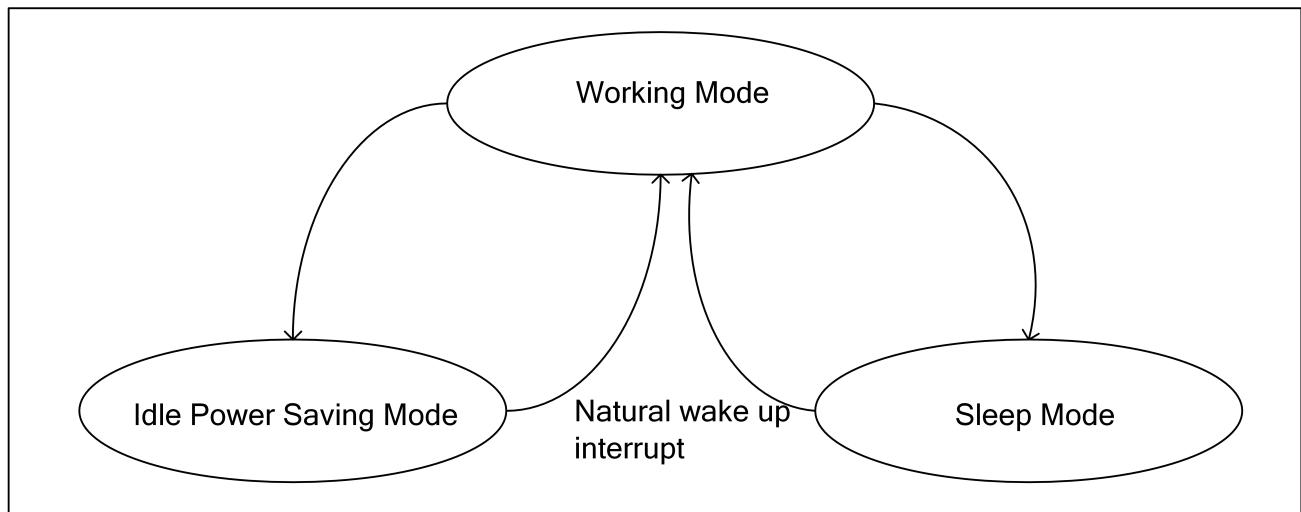


图 33.1: 低功耗模式

### 33.2 主要特征

- 时钟控制:GLB 中对各外设的时钟控制，小范围的省电，响应速度较快
- 睡眠控制 (PDS) : 包含 PDS1/2/3/7 这 4 个等级，大范围省电，响应速度中等
- 深度睡眠控制 (HBN) : 包含 HBN0/1/2/3 这 4 个等级，全局省电，响应时间长

## 33.3 功能描述

### 33.3.1 电源域

BL808 芯片中有 7 个电源域，每个电源域的主要功能如下所示：

- PD\_AON\_RTC
  - 保留 RC32K/XTAL32K 控制寄存器
  - RTC Counter 时钟源选择功能
  - RTC 可以用于唤醒，也可用于 LED 闪烁
- PD\_AON
  - HBN 状态机控制电源/隔离/复位/时钟
  - 保持内部电压输出选择
  - AON\_PIN (GPIO9/10/11/12/13/40/41) 引脚唤醒控制
  - HBN\_OUT0\_PIR (Acomp0/Acomp1/bor/pir) 唤醒屏蔽和使能寄存器、中断状态寄存器
  - BOR (Brown Out Reset) 设置功能
- PD\_AON\_HBNCORE
  - 部分电源控制寄存器
  - 4KB 的 HBN\_RAM，用于在进入 PDS/HBN 模式前保存程序数据，进入 PDS/HBN 后数据不会消失
  - PIR 数字控制，PIR 是热释电红外传感器，HBN 区域内的一个外设，可以用作 HBN 唤醒源
  - AON\_PIN 控制和在 HBN 或 PDS 模式下 IO 保持功能
  - Acomp0、Acomp1 配置，GPADC 时钟源选择和配置
- PD\_CORE
  - HBN 状态机控制电源/隔离/复位/时钟
  - 64KB 保留 RAM
  - WIFI/BLE 计时器控制
  - 160KB WRAM/EM
- PD\_CORE\_MISC\_DIG
  - M0 CPU 及其外设
  - 芯片全局寄存器
- PD\_USB
  - USB 数字控制器
- PD\_MM
  - D0 CPU 及其外设、PSRAM
  - Codec
  - VRAM 32~96KB
  - BLAI

每个电源域都由 9 个不同的电源模式控制，具体的控制方式如下表所示：

表 33.1: 电源模式

NO.	Scenario	Power Domain						
		PD_AON_- RTC	PD_AON	PD_AON_- HBNCORE	PD_- CORE	PD_CORE_MISC_- DIG	PD_USB	PD_MM
1	Normal	ON	ON	ON	ON	ON	ON	ON
2	PDS1	ON	ON	ON	ON	ON	ON	OFF
3	PDS2	ON	ON	ON	ON	ON	OFF	ON
4	PDS3	ON	ON	ON	ON	ON	OFF	OFF
5	PDS7	ON	ON	ON	ON	OFF	OFF	OFF
6	HBN0	ON	ON	ON	OFF	OFF	OFF	OFF
7	HBN1	ON	ON	OFF	OFF	OFF	OFF	OFF
8	HBN2	ON	OFF	OFF	OFF	OFF	OFF	OFF
9	HBN3	OFF	OFF	OFF	OFF	OFF	OFF	OFF

注:

- 进入 HBN2 模式需要芯片先进入 HBN1 后，再断开 VDDIO2 电。
- 在 HBN2 模式下只保留 RTC\_Timer，但是不做唤醒；退出 HBN2 模式只需重新供 VDDIO2。
- VDD33\_RTC/VDDIO2(AON) 共用 pin 的封装 (QFN88 Type2, QFN68) 没有 HBN2 模式。
- 进入 HBN3 模式需要断电 pu\_chip，HBN3 模式将关闭大部分电源，仅 RTC 电源可以通过 rtc\_pu\_chip\_sel 选择是否关掉 RTC 电源。

rtc\_pu\_chip\_sel 为 1，RTC 电源一直保持； rtc\_pu\_chip\_sel 为 0，RTC 电源被 pu\_chip 控制。 rtc\_pu\_chip\_sel 值在封装时决定。

### 33.3.2 唤醒源

芯片内部支持多种唤醒源，可从不同电源模式中唤醒。

PDS1/2/3 可以通过以下方式唤醒:

- HBN 唤醒源
- 所有 GPIO 唤醒
- 红外接收器
- BLE 唤醒事件
- WIFI 唤醒事件
- PDS 计时器

其余电源模式的唤醒源如下表所示:

表 33.2: 唤醒源

电源模式	唤醒源
PDS7	PDS 计时器/PSD_PIN/RTC/AON_PIN/BOR/Pir/Acomp0/Acomp1
HBN0	RTC/AON_PIN/BOR/Pir/Acomp0/Acomp1
HBN1	RTC/AON_PIN
HBN2	重新供 VDDIO2
HBN3	重新供 pu_chip

### 33.3.3 功耗模式

#### 工作模式

芯片提供处理器与外设独立的时钟控制，在 GLB 和时钟的章节介绍对各模块的时钟控制，软件可以根据当前应用场景，对于不需要使用的处理器或外设进行时钟控制。时钟控制的反应是实时的，在此工作模式下，不需要担心响应时间。

#### 掉电睡眠模式 (PDS)

掉电模式相较于工作模式功耗较低。进入 PDS 模式后，将 RTC(Real Time Clock) 之外的时钟进行管控，会切换为内部低速时钟，并将外部晶振与 PLL 关闭达到更加省电的状态，因此进入与离开此低功耗模式会有时间延迟。当进入掉电睡眠模式时，OCRAM 区域的数据可以自动进入 retention 状态而保留下来，当唤醒后可以自行退出 retention 状态。

##### 1. 进入空闲省电模式

软件可通过 PDS 配置让此模块进入掉电模式，等待处理，进入等待中断模式 (WFI) 后，PDS 模块会触发时钟控制模块进入 gate clock 操作，并通知模拟电路关闭 PLL 以及外部晶振

##### 2. 离开空闲省电模式

离开空闲省电模式的方式有两种，第一是空闲中间有特定的中断或事件打断空闲状态，第二是软件设定 PDS\_TIM 的时间达到，两者均会触发 PDS 模块离开掉电模式。注意：因为打开晶振需要约 1ms 的时间，PDS 提供软件提前打开晶振的方式，这个做法可以加速 PDS 醒来。当 PDS 模块准备醒来时，此模块会通过中断通知处理器离开等待中断模式 (WFI)。

#### 休眠模式 (HBN)

休眠模式在保留 AON(Always On) 电源的状态下，将大部分的芯片逻辑进行断电 (Vcore)，直到收到外部事件才会将内部电路唤醒的。在休眠模式下可以达到极致的省电状态，但相对于前两者需要的响应时间也最长，适合长时间不需要工作的状态下，可以进入此状态，延长电池寿命。休眠时期会将大部分的电路断电，对应的寄存器值和内存的数据也会消失。因此 HBN 内部留有 4KB HBN\_RAM，这个内存 HBN0 模式时不会断电，软件有需要保存的资料或状态可以在进入休眠前拷贝到这个内存。从休眠恢复时，可以直接从 RAM 中存取数据，通常可以用作状态的纪录或是数据快速恢复。

### 33.3.4 IO 保持

IO 保持可以分为 AON\_IO 保持和 PDS\_IO 保持。PDS0/1/2/3 模式下，由于芯片 MISC domain 仍有电，所以 GPIO 可被 glb 寄存器控制。当 glb 寄存器被断电之后，可由 AON\_CTRL 和 PDS 简单控制 AON\_IO 和 PDS\_IO 的 IE/PD/PU。

#### AON\_IO

AON\_IO 指 GPIO9/10/11/12/13/14/15/40/41。GPIO40/41 默认作为 XTAL32K 输入和输出使用（要改成其它 pinmux 用途才可以作为 AON\_IO 使用）当 GPIO40/41 的 IE/OE 都被设为 0 时，GPIO40/41 复用模拟功能 XTAL32K\_INXTAL32K\_OUT；相反，当 GPIO40/41 的 IE/OE 不都为 0 时，GPIO40/41 用作普通 IO 功能。例如，GPIO40 作为普通 IO 功能具体配置如下：reg\_en\_aon\_ctrl\_gpio[7] 和 reg\_aon\_pad\_oe[7] 配为 0，reg\_aon\_pad\_ie\_smt[7] 配为 1。

1、硬件 IO 保持 HBN 可以控制 AON\_IO 的 IE/PD/PU/OE/O，从而实现 IO 保持。reg\_en\_aon\_ctrl\_gpio 为 1 时，AON\_IO 的上拉使能由 reg\_aon\_gpio\_pu 控制，下拉使能由 reg\_aon\_gpio\_pd 控制，OE 由 reg\_aon\_gpio\_oe 控制，AON PAD O 分别由 aon\_led\_out[0]、[1]、[2] 控制。而 IE/SMT，无论 reg\_en\_aon\_ctrl\_gpio 的值是 0 还是 1，都可以由 reg\_aon\_gpio\_ie\_smt 控制。例如，reg\_en\_aon\_ctrl\_gpio 为 0，即使 reg\_aon\_pad\_pu 为 1，也不能实现上拉，但是 reg\_aon\_gpio\_ie\_smt 为 1，可以实现 IE 功能。

2、软件 IO 保持将 reg\_aon\_gpio\_iso\_mode 配置为 1 后，进入 HBN 模式时，AON PAD 可以保持 OEO，PUPD 不能保持；等到 HBN 唤醒后，AON PAD 状态仍会保持着，需要把 reg\_aon\_gpio\_iso\_mode 清 0 后，才会离开 IO 保持状态。例如，GPIO40 在 HBN 模式下保持高电平，需要先将其配为普通 IO 功能，然后用 glb 寄存器配置为输出高电平，最后将 reg\_aon\_gpio\_iso\_mode 配置为 1 后，进入 HBN 模式。

#### PDS\_IO

PDS\_IO 指除 AON\_IO 以外的其他 GPIO，共 32 个 GPIO，按 PAD 的物理位置分为 3 组：

- 左面: GPIO0~8
- 右面: GPIO16~23
- 上面: GPIO24~39

1、硬件 IO 保持 PDS\_IO 的 IE/PD/PU 可由 pds\_gpio\_i\_set 寄存器控制，相同的组 GPIO 必须保持相同的电平。例如，GPIO0 是配置为上拉，那么 GPIO8 也是配置为上拉的。

2、软件 IO 保持当 cr\_pds\_gpio\_iso\_mode 为 1 时，进入 PDS7 模式后，如果 cr\_pds\_gpio Kee\_en[0]、[1]、[2] 为 1，GPIO0~8、GPIO16~23、GPIO24~38 分别进入 GPIO 保持状态，等到 PDS7 唤醒后，PDS\_IO 状态仍会保持着，需要把 cr\_pds\_gpio\_iso\_mode 清 0 后，才会离开 IO 保持状态。这种 IO 保持方式的优势是可以实现同一组 GPIO 保持不同的电平。

## 33.4 寄存器描述

名称	描述
HBN_TIME_L	
HBN_TIME_H	

### 33.4.1 HBN\_TIME\_L

地址: 0x2000f004

hbn\_time\_l

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

hbn\_time\_l

位	名称	权限	复位值	描述
31:0	hbn_time_l	r/w	32'h0	RTC timer compare bit 31:0

### 33.4.2 HBN\_TIME\_H

地址: 0x2000f008

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

hbn\_time\_h

位	名称	权限	复位值	描述
31:8	RSVD			
7:0	hbn_time_h	r/w	8'h0	RTC timer compare bit 39:32

## 34.1 简介

### 34.1.1 AES 简介

高级加密标准 (AES, Advanced Encryption Standard) 为最常见的对称加密算法。在密码学中又称 Rijndael 加密法，是美国联邦政府采用的一种区块加密标准。

### 34.1.2 MD5 简介

MD5 信息摘要算法（英语：MD5 Message-Digest Algorithm），一种被广泛使用的密码散列函数，可以产生出一个 128 位（16 字节）的散列值（hash value），用于确保信息传输完整一致。

### 34.1.3 SHA 简介

SHA256 是 SHA-2 下细分出的一种算法, SHA-2，名称来自于安全散列算法 2（英语：Secure Hash Algorithm 2）的缩写，一种密码散列函数算法标准，由美国国家安全局研发，属于 SHA 算法之一，是 SHA-1 的后继者。SHA-2 下又可再分为六个不同的算法标准，包括了：SHA-224、SHA-256、SHA-384、SHA-512、SHA-512/224、SHA-512/256。其中 SHA-512/224 指结果取 SHA-512 的前 224 个 bit，SHA-512/256 指结果取 SHA-512 的前 256 个 bit。

### 34.1.4 CRC 简介

循环冗余校验（Cyclic Redundancy Check, CRC）是一种根据网络数据包或计算机文件等数据产生简短固定位数校验码的一种信道编码技术，主要用来检测或校验数据传输或者保存后可能出现的错误。它是利用除法及余数的原理来作错误侦测的。

### 34.1.5 GMAC 简介

GMAC 就是利用伽罗华域 (Galois Field, GF, 有限域) 乘法运算来计算消息的 MAC 值。

## 34.2 主要特征

- 支持 aes-128, aes-192, aes-256 加解密
- 支持 crc-16, crc-32, md5, sha-256, sha-512
- 支持 trng
- 支持 gmac

## 34.3 原理描述

AES 对称加密算法也就是加密和解密用相同的密钥，加密流程如下图：

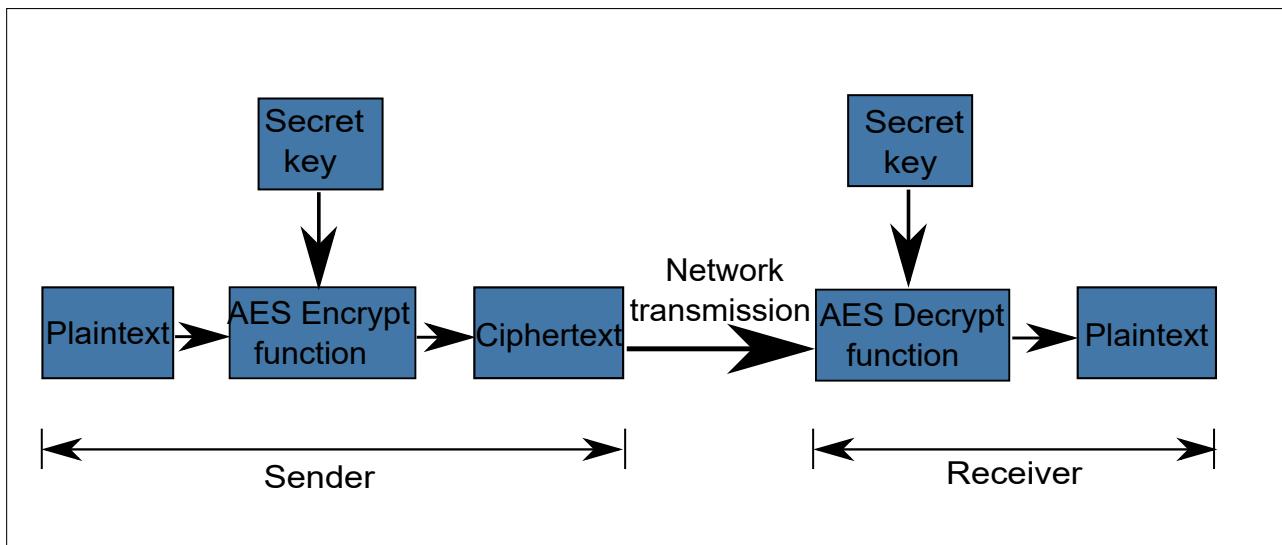


图 34.1: AES 加密流程图

下面简单介绍下各个部分的作用与意义：

- 明文 P: 没有经过加密的数据。
- 密钥 K: 用来加加密明文的密码，在对称加密算法中，加密与解密的密钥是相同的。密钥为接收方与发送方协商产生，但不可以直接在网络上传输，否则会导致密钥泄漏，通常是通过非对称加密算法加密密钥，然后再通过网络传输给对方，或者直接面对面商量密钥。密钥是绝对不可以泄漏的，否则会被攻击者还原密文，窃取机密数据。
- AES 加密函数：设 AES 加密函数为 E，则  $C = E(K, P)$ , 其中 P 为明文，K 为密钥，C 为密文。也就是说，把明文 P 和密钥 K 作为加密函数的参数输入，则加密函数 E 会输出密文 C。
- 密文 C: 经加密函数处理后的数据。
- AES 解密函数: 设 AES 解密函数为 D, 则  $P = D(K, C)$ , 其中 C 为密文，K 为密钥，P 为明文。也就是说，把密文 C 和密钥 K 作为解密函数的参数输入，则解密函数会输出明文 P。

### 34.3.1 MD5 的实现：

MD5 可以认为是基于 block 的算法：它要求每次处理的数据长度为 512bits。但是实际中要处理的明文长度并不一定是 512 的整数倍，需要做数据补齐/填充（Padding）。假设原始明文消息的长度为 K，MD5 的 Padding 可以细分为 2 个子步骤：

- 1. 附加填充位（Append Padding Bits）：从原始明文消息的 K 位之后补 100... 一直到  $512-64=448$  位，填充位的规则是：只有第一个 bit 是 1，之后都是 0。
- 2. 附加长度（Append Length）：在第一步结果之后再填充上原消息的长度，可用来进行的存储长度为 64 位。整个 Padding 的示意图如下所示：

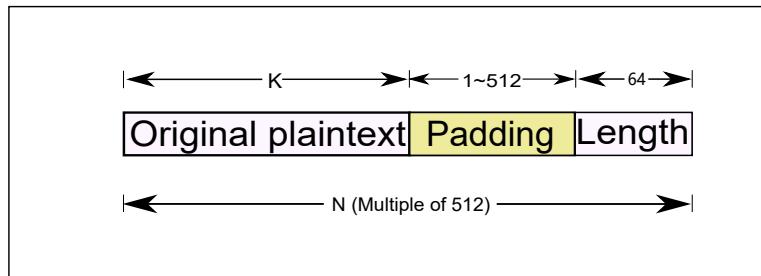


图 34.2: MD5 Padding

准备需要用到的数据：

- 4 个常数：A = 0x67452301, B = 0x0EFCDAB89, C = 0x98BADCCE, D = 0x10325476;
- 4 个函数：F(X,Y,Z)=(X & Y) | ((~X) & Z); G(X,Y,Z)=(X & Z) | (Y & (~Z)); H(X,Y,Z)=X ^ Y ^ Z; I(X,Y,Z)=Y ^ (X | (~Z));

把消息分以 512 位为一分组进行处理，每一个分组进行 4 轮变换，以上面所说 4 个常数为起始变量进行计算，重新输出 4 个变量，以这 4 个变量再进行下一分组的运算，如果已经是最后一个分组，则这 4 个变量为最后的结果，即 MD5 值。

### 34.3.2 SHA256 的实现：

和 MD5 有些类似，数据填充遵循的准则和 MD5 一样。SHA256 算法中用到了 8 个哈希初值以及 64 个哈希常量。

其中，SHA256 算法的 8 个哈希初值如下：

- h0 := 0x6a09e667
- h1 := 0xbb67ae85
- h2 := 0x3c6ef372
- h3 := 0xa54ff53a
- h4 := 0x510e527f
- h5 := 0x9b05688c
- h6 := 0x1f83d9ab
- h7 := 0x5be0cd19

这些初值是对自然数中前 8 个质数（2,3,5,7,11,13,17,19）的平方根的小数部分取前 32bit 而来。

在 SHA256 算法中，用到的 64 个常量如下：

- 428a2f98 71374491 b5c0fbcf e9b5dba5
- 3956c25b 59f111f1 923f82a4 ab1c5ed5
- d807aa98 12835b01 243185be 550c7dc3
- 72be5d74 80deb1fe 9bdc06a7 c19bf174
- e49b69c1 efbe4786 0fc19dc6 240ca1cc
- 2de92c6f 4a7484aa 5cb0a9dc 76f988da
- 983e5152 a831c66d b00327c8 bf597fc7
- c6e00bf3 d5a79147 06ca6351 14292967
- 27b70a85 2e1b2138 4d2c6dfc 53380d13
- 650a7354 766a0abb 81c2c92e 92722c85
- a2bfe8a1 a81a664b c24b8b70 c76c51a3
- d192e819 d6990624 f40e3585 106aa070
- 19a4c116 1e376c08 2748774c 34b0bcb5
- 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
- 748f82ee 78a5636f 84c87814 8cc70208
- 90beffff a4506ceb bef9a3f7 c67178f2

和 8 个哈希初值类似，这些常量是对自然数中前 64 个质数 (2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83, 89,97...) 的立方根的小数部分取前 32bit 而来。

SHA256 散列函数：

- $Ch(x,y,z) = (x \square y) \square (\neg x \square z)$
- $Ma(x,y,z) = (x \square y) \square (x \square z) \square (y \square z)$
- $\Sigma 0(x) = S^2(x) \square S^{13}(x) \square S^{22}(x)$
- $\Sigma 1(x) = S^6(x) \square S^{11}(x) \square S^{25}(x)$
- $\sigma 0(x) = S^7(x) \square S^{18}(x) \square R^3(x)$
- $\sigma 1(x) = S^{17}(x) \square S^{19}(x) \square R^{10}(x)$

其中：

- $\square$  按位“与”
- $\neg$  按位“补”
- $\square$  按位“异或”
- $S^n$  循环右移 n 个 bit
- $R^n$  右移 n 个 bit

### 34.3.3 GMAC 的原理

消息认证实际上是对消息本身产生的一个冗余的信息，即消息验证码（MAC）。消息认证码（Message authentication code）是一种确认完整性并进行认证的一种技术，简称 MAC。密码学中，消息认证码指的是通信实体双方使用的一种验证机制，保证消息数据完整性的一种工具。消息认证码是一种带密钥的哈希函数，它本质上是一个哈希函数，那为什么要带密钥呢？是因为消息在传输过程中是可以被篡改，哈希值也可以被篡改，因此为了保证这个哈希值的有效性，通过加密的方式将哈希值保护起来，这样在接收方接收到消息后就可以通过这个哈希值来判断整条消息的完整性，从而达到信息传递的目的。

消息认证码步骤如下图所示：

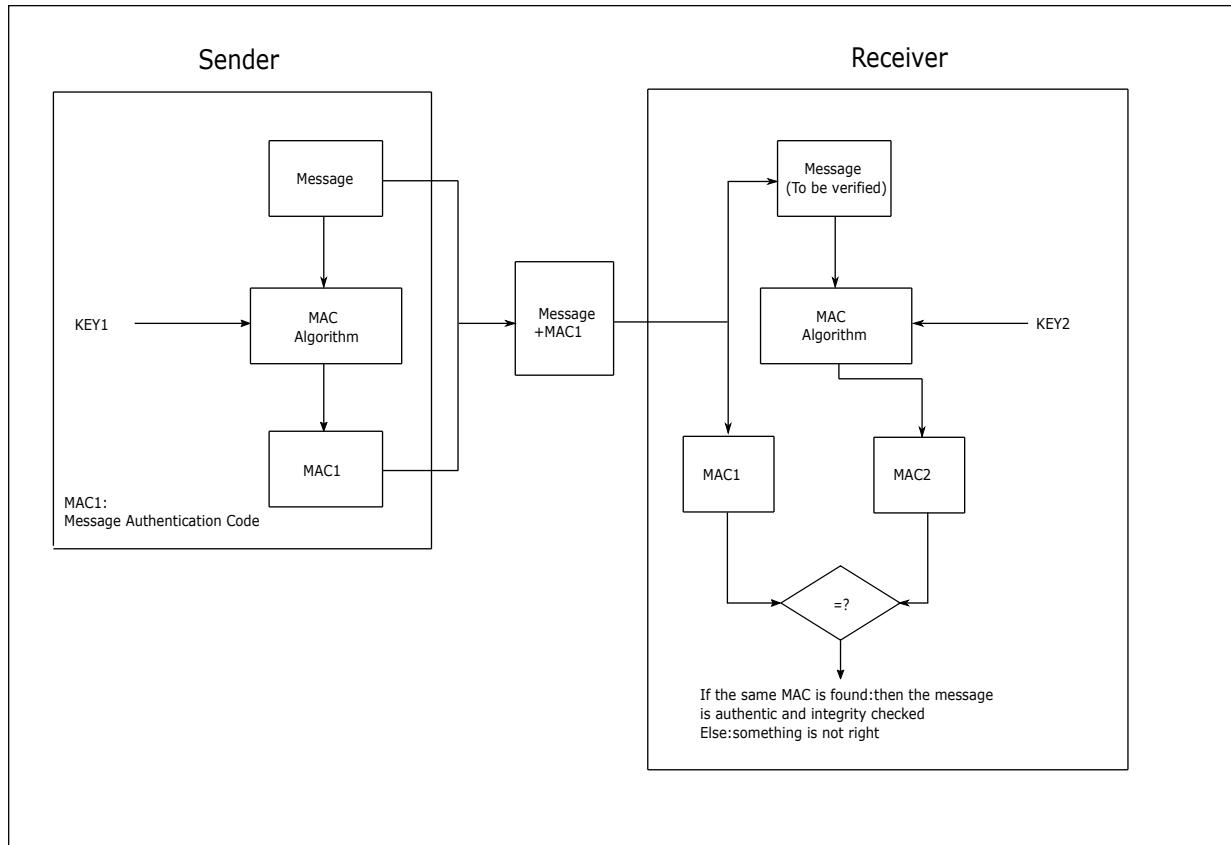


图 34.3: 消息认证码流程图

- 1) 发送者与接收者事先共享密钥 K (上图中的 KEY1 与 KEY2 值保持一致)。
- 2) 发送者根据消息计算 MAC 值 (使用密钥 KEY1 对原始消息计算 MAC1)。
- 3) 发送者将原始消息和 MAC1 发送给接收者。
- 4) 接收者根据收到的原始消息计算 MAC2 (使用密钥 KEY2)。
- 5) 接收者将自己计算出的 MAC2 与从发送者收到的 MAC1 比对。
- 6) 如果 MAC 一致，接收者可以判定消息的确来自接收者 (认证成功) 且没有被篡改或者出现传输出错的情况；如果不一致，可判断消息不是来自发送方 (认证失败)。

注意：建议发送方和接收方将密钥 KEY 存放于硬件安全模块中，计算 MAC 值的过程最好也放到硬件安全模块中完成，这样可以保证密钥的安全，例如放到加密芯片中。

GMAC 就是利用伽罗华域 (Galois Field, GF, 有限域) 乘法运算来计算消息的 MAC 值。

## 34.4 功能描述

### 34.4.1 AES 加速器

1 AES 加速器支持 AES-128/192/256 加解密 6 种运算。

- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_mode 和 se\_aes\_0\_dec\_en, 如下图所示:

amode	adec en	运算
0	0	AES-128 加密
1	0	AES-256 加密
2	0	AES-192 加密
0	1	AES-128 解密
1	1	AES-256 解密
2	1	AES-192 解密

图 34.4: AES 运算模式图

配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_block\_mode 选择不同的加密方式，目前支持 ECB、CTR、CBC、XTS 模式。

### 2 密钥、明文、密文、初始化向量

- 寄存器 se\_aes\_0\_msa 存放明文或者密文的地址。
- 寄存器 se\_aes\_0\_mda 存放密文或者明文的地址。
- 寄存器 se\_aes\_0\_iv\_0~se\_aes\_0\_iv\_3 存放 IV。
- 寄存器 se\_aes\_0\_key\_0~se\_aes\_0\_key\_7 存放密钥。

### 3 软硬件加密流程

- 配置寄存器 se\_aes\_0\_endian, 其中包括 se\_aes\_0\_dout\_endian、se\_aes\_0\_din\_endian、se\_aes\_0\_key\_endian、se\_aes\_0\_iv\_endian、se\_aes\_0\_twk\_endian  
备注: 0:little-endian 1:big-endian
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_block\_mode
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_mode
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_dec\_en
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_dec\_key\_sel 0:new key 1:same key as last one
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_iv\_sel 0:new iv 1:same iv as last one

- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_en enable aes
- 配置寄存器 se\_aes\_0\_iv\_0~se\_aes\_0\_iv\_3 set IV。  
填写顺序备注: MSB: se\_aes\_0\_iv\_0~se\_aes\_0\_iv\_3; LSB: se\_aes\_0\_iv\_3~se\_aes\_0\_iv\_0
- 配置寄存器 se\_aes\_0\_key\_0~se\_aes\_0\_key\_7 set key。  
填写顺序备注: MSB: se\_aes\_0\_key\_0~se\_aes\_0\_key\_7; LSB: se\_aes\_0\_key\_7~se\_aes\_0\_key\_0。  
填写个数备注: AES-128 取前 4 个, AES-196 取前 6 个, AES-256 取 8 个。
- 配置寄存器 se\_aes\_0\_msa set msa addr
- 配置寄存器 se\_aes\_0\_mda set mda addr
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_msg\_len set msg len
- 配置寄存器 se\_aes\_0\_ctrl 中 se\_aes\_0\_trig\_1t Trigger AES Engine
- 结果输出存储在寄存器 se\_aes\_0\_mda 对应的地址中

#### 34.4.2 SHA 加速器

1 SHA 加速器支持 7 种标准运算:

- SHA-1、SHA-224、SHA-256、SHA-512、SHA-384、SHA-512/224、SHA-512/256, 同时还支持 MD5、CRC16、CRC32。

寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_mode: 0:SHA-256 1:SHA-224 2:SHA-1 3:SHA-14:SHA-512 5:SHA-384 6:SHA-512/224 7:SHA-512/256

寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_mode\_ext: hash mode extention; 0:SHA 1:MD5 2:CRC-16 3:CRC-32

配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_mode 选择不同的 SHA 运算, 配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_mode\_ext 可选择 MD5、CRC16、CRC32。

- 当 se\_sha\_0\_mode\_ext 为 0 时, se\_sha\_0\_mode 有效。
- 当 se\_sha\_0\_mode\_ext 不为 0 时, se\_sha\_0\_mode 无效。

2 明文、密文

- 寄存器 se\_sha\_0\_msa 存放明文地址。
- 寄存器 se\_sha\_0\_hash\_l\_0~se\_sha\_0\_hash\_l\_7 存放密文。

获取顺序备注: MSB:se\_sha\_0\_hash\_l\_0~se\_sha\_0\_hash\_l\_7; LSB:se\_sha\_0\_hash\_l\_7~se\_sha\_0\_hash\_l\_0

3 运算流程

- 配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_mode set SHA operation type
- 配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_en enable sha
- 配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_hash\_sel 0:new hash 1:accumulate last hash
- 配置寄存器 se\_sha\_0\_msa set mda addr
- 配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_msg\_len set msg len
- 配置寄存器 se\_sha\_0\_ctrl 中 se\_sha\_0\_trig\_1t Trigger SHA Engine
- 结果输出存储在 se\_sha\_0\_hash\_l\_0~se\_sha\_0\_hash\_l\_7 MSB:se\_sha\_0\_hash\_l\_0~se\_sha\_0\_hash\_l\_7 LSB:se\_sha\_0\_hash\_l\_7~se\_sha\_0\_hash\_l\_0

### 34.4.3 随机数发生器

1. 内置一个真随机数发生器，其生成的随机数可作为加密等操作的基础。

- 真随机数：真正的随机数是使用物理现象产生的：比如掷钱币、骰子、转轮、使用电子元件的噪音、核裂变等等，这样的随机数发生器叫做物理性随机数发生器，它们的缺点是技术要求比较高。
- 伪随机数：真正意义上的随机数（或者随机事件）在某次产生过程中是按照实验过程中表现的分布概率随机产生的，其结果是不可预测的，是不可见的。而计算机中的随机函数是按照一定算法模拟产生的，其结果是确定的，是可见的。我们可以这样认为这个可预见的结果其出现的概率是 100%。所以用计算机随机函数所产生的“随机数”并不随机，是伪随机数。

2. 输出

- 寄存器 SE\_TRNG\_0\_DOUT\_0~SE\_TRNG\_0\_DOUT\_7 存放输出的随机数。

3. 使用流程

- 配置寄存器 se\_trng\_0\_ctrl\_0 中 se\_trng\_0\_en enable trng
- 配置寄存器 se\_trng\_0\_ctrl\_0 中 se\_trng\_0\_trig\_1t trigger trng engine
- 结果输出存储在 se\_trng\_0\_dout\_0~se\_trng\_0\_dout\_7

### 34.4.4 GMAC(link 模式)

1.GMAC\_Link\_Table 结构体定义

- Word0:
  - [9]:se\_gmac\_0\_int\_clr\_1t
  - [10]:se\_gmac\_0\_int\_set\_1t
  - [31:16]:se\_gmac\_0\_msg\_len
- Word1:se\_gmac\_0\_msa
- Word2、Word3、Word4、Word5: se\_gmac\_0\_h
- Word6、Word7、Word8、Word9: se\_gmac\_0\_tag

2. 使用流程

- 配置寄存器 se\_gmac\_0\_ctrl\_0 中 se\_gmac\_0\_x\_endian、se\_gmac\_0\_h\_endian、se\_gmac\_0\_t\_endian 0:little-endian 1:big-endian
- 将 GMAC\_Link\_Table 结构体的起始地址写到寄存器 se\_gmac\_0\_lca 中
- 将原始消息地址赋给 GMAC\_Link\_Table 结构体中 Word1 中 se\_gmac\_0\_msa
- 将原始消息长度赋给 GMAC\_Link\_Table 结构体中 Word0 中 se\_gmac\_0\_msg\_len, 128bit 消息长度对应 se\_gmac\_0\_msg\_len 中的 1
- 配置寄存器 se\_gmac\_0\_ctrl\_0 中 se\_gmac\_0\_trig\_1t trigger gmac engine
- 结果输出存储在 GMAC\_Link\_Table 结构体中 Word6~Word9

## 34.5 寄存器描述

名称	描述
se_sha_0_ctrl	
se_sha_0_msa	
se_sha_0_status	
se_sha_0_endian	
se_sha_0_hash_l_0	
se_sha_0_hash_l_1	
se_sha_0_hash_l_2	
se_sha_0_hash_l_3	
se_sha_0_hash_l_4	
se_sha_0_hash_l_5	
se_sha_0_hash_l_6	
se_sha_0_hash_l_7	
se_sha_0_hash_h_0	
se_sha_0_hash_h_1	
se_sha_0_hash_h_2	
se_sha_0_hash_h_3	
se_sha_0_hash_h_4	
se_sha_0_hash_h_5	
se_sha_0_hash_h_6	
se_sha_0_hash_h_7	
se_sha_0_link	
se_sha_0_ctrl_prot	
se_aes_0_ctrl	
se_aes_0_msa	
se_aes_0_mda	
se_aes_0_status	
se_aes_0_iv_0	
se_aes_0_iv_1	
se_aes_0_iv_2	
se_aes_0_iv_3	

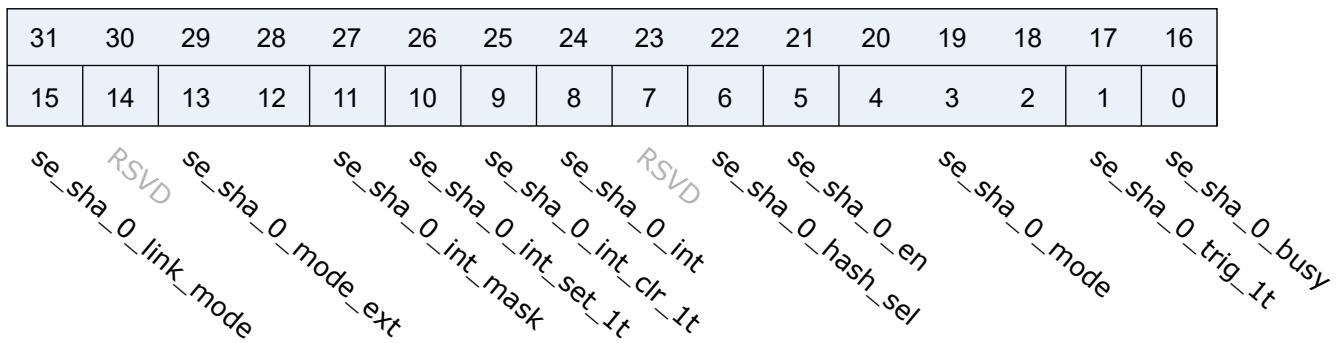
名称	描述
se_aes_0_key_0	
se_aes_0_key_1	
se_aes_0_key_2	
se_aes_0_key_3	
se_aes_0_key_4	
se_aes_0_key_5	
se_aes_0_key_6	
se_aes_0_key_7	
se_aes_0_key_sel	
se_aes_1_key_sel	
se_aes_0_endian	
se_aes_sboot	
se_aes_0_link	
se_aes_0_ctrl_prot	
se_trng_0_ctrl_0	
se_trng_0_status	
se_trng_0_dout_0	
se_trng_0_dout_1	
se_trng_0_dout_2	
se_trng_0_dout_3	
se_trng_0_dout_4	
se_trng_0_dout_5	
se_trng_0_dout_6	
se_trng_0_dout_7	
se_trng_0_test	
se_trng_0_ctrl_1	
se_trng_0_ctrl_2	
se_trng_0_ctrl_3	
se_trng_0_test_out_0	
se_trng_0_test_out_1	
se_trng_0_test_out_2	

名称	描述
se_trng_0_test_out_3	
se_trng_0_ctrl_prot	
se_pka_0_ctrl_0	
se_pka_0_seed	
se_pka_0_ctrl_1	
se_pka_0_rw	
se_pka_0_rw_burst	
se_pka_0_ctrl_prot	
se_cdet_0_ctrl_0	
se_cdet_0_ctrl_1	
se_cdet_0_ctrl_prot	
se_gmac_0_ctrl_0	
se_gmac_0_lca	
se_gmac_0_status	
se_gmac_0_ctrl_prot	
se_ctrl_prot_rd	

### 34.5.1 se\_sha\_0\_ctrl

地址: 0x20004000

se\_sha\_0\_msg\_len



位	名称	权限	复位值	描述
31:16	se_sha_0_msg_len	r/w	0	number of 512-bit block

位	名称	权限	复位值	描述
15	se_sha_0_link_mode	r/w	0	1:enable sha link mode
14	RSVD			
13:12	se_sha_0_mode_ext	r/w	0	hash mode extention; 0:SHA 1:MD5 2:CRC-16 3:CRC-32
11	se_sha_0_int_mask	r/w	0	
10	se_sha_0_int_set_1t	w1p	0	1:set interrupt
9	se_sha_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_sha_0_int	r	0	interrupt value
7	RSVD			
6	se_sha_0_hash_sel	r/w	0	0:new hash 1:accumulate last hash
5	se_sha_0_en	r/w	0	1:enable sha engine
4:2	se_sha_0_mode	r/w	0	0:SHA-256 1:SHA-224 2:SHA-1 3:SHA-1 4:SHA-512 5:SHA-384 6:SHA-512/224 7:SHA-512/256
1	se_sha_0_trig_1t	w1p	0	1:trigger sha engine
0	se_sha_0_busy	r	0	1:sha engine busy

### 34.5.2 se\_sha\_0\_msa

地址: 0x20004004

se\_sha\_0\_msa

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_msa

位	名称	权限	复位值	描述
31:0	se_sha_0_msa	r/w	0	message source address

### 34.5.3 se\_sha\_0\_status

地址: 0x20004008

se\_sha\_0\_status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_status

位	名称	权限	复位值	描述
31:0	se_sha_0_status	r	32'h41	

#### 34.5.4 se\_sha\_0\_endian

地址: 0x2000400c

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
14	RSVD														
13	RSVD														
12	RSVD														
11	RSVD														
10	RSVD														
9	RSVD														
8	RSVD														
7	RSVD														
6	RSVD														
5	RSVD														
4	RSVD														
3	RSVD														
2	RSVD														
1	RSVD														
0	RSVD														

位	名称	权限	复位值	描述
31:1	RSVD			
0	se_sha_0_dout_endian	r/w	1	0:little-endian 1:big-endian

#### 34.5.5 se\_sha\_0\_hash\_l\_0

地址: 0x20004010

se_sha_0_hash_l_0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
14	RSVD														
13	RSVD														
12	RSVD														
11	RSVD														
10	RSVD														
9	RSVD														
8	RSVD														
7	RSVD														
6	RSVD														
5	RSVD														
4	RSVD														
3	RSVD														
2	RSVD														
1	RSVD														
0	RSVD														

se\_sha\_0\_hash\_l\_0

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_0	r	0	big-endian hash 0 (MSB)

### 34.5.6 se\_sha\_0\_hash\_l\_1

地址: 0x20004014

se\_sha\_0\_hash\_l\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_1

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_1	r	0	big-endian hash 1

### 34.5.7 se\_sha\_0\_hash\_l\_2

地址: 0x20004018

se\_sha\_0\_hash\_l\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_2

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_2	r	0	big-endian hash 2

### 34.5.8 se\_sha\_0\_hash\_l\_3

地址: 0x2000401c

se\_sha\_0\_hash\_l\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_3

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_3	r	0	big-endian hash 3

### 34.5.9 se\_sha\_0\_hash\_l\_4

地址: 0x20004020

se\_sha\_0\_hash\_l\_4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_4

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_4	r	0	big-endian hash 4

### 34.5.10 se\_sha\_0\_hash\_l\_5

地址: 0x20004024

se\_sha\_0\_hash\_l\_5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_5

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_5	r	0	big-endian hash 5

### 34.5.11 se\_sha\_0\_hash\_l\_6

地址: 0x20004028

se\_sha\_0\_hash\_l\_6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_6

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_6	r	0	big-endian hash 6

### 34.5.12 se\_sha\_0\_hash\_l\_7

地址: 0x2000402c

se\_sha\_0\_hash\_l\_7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_l\_7

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_l_7	r	0	big-endian hash 7 (LSB)

### 34.5.13 se\_sha\_0\_hash\_h\_0

地址: 0x20004030

se\_sha\_0\_hash\_h\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_0

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_0	r	0	big-endian hash 0 (MSB)

### 34.5.14 se\_sha\_0\_hash\_h\_1

地址: 0x20004034

se\_sha\_0\_hash\_h\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_1

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_1	r	0	big-endian hash 1

### 34.5.15 se\_sha\_0\_hash\_h\_2

地址: 0x20004038

se\_sha\_0\_hash\_h\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_2

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_2	r	0	big-endian hash 2

### 34.5.16 se\_sha\_0\_hash\_h\_3

地址: 0x2000403c

se\_sha\_0\_hash\_h\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_3

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_3	r	0	big-endian hash 3

### 34.5.17 se\_sha\_0\_hash\_h\_4

地址: 0x20004040

se\_sha\_0\_hash\_h\_4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_4

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_4	r	0	big-endian hash 4

### 34.5.18 se\_sha\_0\_hash\_h\_5

地址: 0x20004044

se\_sha\_0\_hash\_h\_5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_5

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_5	r	0	big-endian hash 5

### 34.5.19 se\_sha\_0\_hash\_h\_6

地址: 0x20004048

se\_sha\_0\_hash\_h\_6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_6

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_6	r	0	big-endian hash 6

### 34.5.20 se\_sha\_0\_hash\_h\_7

地址: 0x2000404c

se\_sha\_0\_hash\_h\_7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_hash\_h\_7

位	名称	权限	复位值	描述
31:0	se_sha_0_hash_h_7	r	0	big-endian hash 7 (LSB)

### 34.5.21 se\_sha\_0\_link

地址: 0x20004050

se\_sha\_0\_lca

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_sha\_0\_lca

位	名称	权限	复位值	描述
31:0	se_sha_0_lca	r/w	0	aes link config address(word align)

### 34.5.22 se\_sha\_0\_ctrl\_prot

地址: 0x200040fc

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

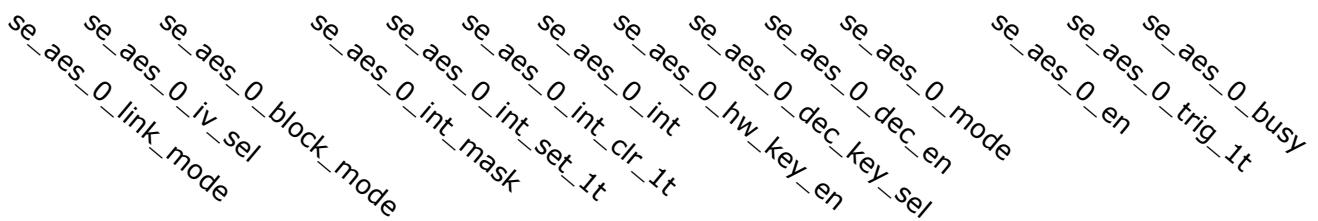
RSVD	se_sha_id0_en	RSVD														
															se_sha_id1_en	

位	名称	权限	复位值	描述
31:3	RSVD			
2	se_sha_id1_en	r/w	1	id1 access right
1	se_sha_id0_en	r/w	1	id0 access right
0	RSVD			

### 34.5.23 se\_aes\_0\_ctrl

地址: 0x20004100

se_aes_0_msg_len															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0



位	名称	权限	复位值	描述
31:16	se_aes_0_msg_len	r/w	0	number of 128-bit block
15	se_aes_0_link_mode	r/w	0	1:enable aes link mode
14	se_aes_0_iv_sel	r/w	0	0:new iv 1:same iv as last one
13:12	se_aes_0_block_mode	r/w	0	0:ECB mode 1:CTR mode 2:CBC mode 3:XTS mode
11	se_aes_0_int_mask	r/w	0	
10	se_aes_0_int_set_1t	w1p	0	1:set interrupt
9	se_aes_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_aes_0_int	r	0	interrupt value
7	se_aes_0_hw_key_en	r/w	0	0:sw key 1:hw key
6	se_aes_0_dec_key_sel	r/w	0	0:new key 1:same key as last one
5	se_aes_0_dec_en	r/w	0	0:encode 1:decode
4:3	se_aes_0_mode	r/w	0	0:128-bit mode 1:256-bit mode 2:192-bit mode 3:128-bit double key mode
2	se_aes_0_en	r/w	0	0:disable 1:enable aes
1	se_aes_0_trig_1t	w1p	0	1:trigger aes engine
0	se_aes_0_busy	r	0	1:aes engine busy

### 34.5.24 se\_aes\_0\_msa

地址: 0x20004104

se_aes_0_msa															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_msa

位	名称	权限	复位值	描述
31:0	se_aes_0_msa	r/w	0	message source address

### 34.5.25 se\_aes\_0\_mda

地址: 0x20004108

se\_aes\_0\_mda

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_mda

位	名称	权限	复位值	描述
31:0	se_aes_0_mda	r/w	0	message destination address

### 34.5.26 se\_aes\_0\_status

地址: 0x2000410c

se\_aes\_0\_status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_status

位	名称	权限	复位值	描述
31:0	se_aes_0_status	r	32'h4100	

### 34.5.27 se\_aes\_0\_iv\_0

地址: 0x20004110

se\_aes\_0\_iv\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_iv\_0

位	名称	权限	复位值	描述
31:0	se_aes_0_iv_0	r/w	0	big endian initial vector (MSB)

### 34.5.28 se\_aes\_0\_iv\_1

地址: 0x20004114

se\_aes\_0\_iv\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_iv\_1

位	名称	权限	复位值	描述
31:0	se_aes_0_iv_1	r/w	0	big endian initial vector

### 34.5.29 se\_aes\_0\_iv\_2

地址: 0x20004118

se\_aes\_0\_iv\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_iv\_2

位	名称	权限	复位值	描述
31:0	se_aes_0_iv_2	r/w	0	big endian initial vector

### 34.5.30 se\_aes\_0\_iv\_3

地址: 0x2000411c

se\_aes\_0\_iv\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_iv\_3

位	名称	权限	复位值	描述
31:0	se_aes_0_iv_3	r/w	0	big endian initial vector (LSB) (CTR mode: 32-bit counter initial value)

### 34.5.31 se\_aes\_0\_key\_0

地址: 0x20004120

se\_aes\_0\_key\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_key\_0

位	名称	权限	复位值	描述
31:0	se_aes_0_key_0	r/w	0	big endian aes key (aes-128/256 key MSB)

### 34.5.32 se\_aes\_0\_key\_1

地址: 0x20004124

se\_aes\_0\_key\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_key\_1

位	名称	权限	复位值	描述
31:0	se_aes_0_key_1	r/w	0	big endian aes key

### 34.5.33 se\_aes\_0\_key\_2

地址: 0x20004128

se\_aes\_0\_key\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_key\_2

位	名称	权限	复位值	描述
31:0	se_aes_0_key_2	r/w	0	big endian aes key

### 34.5.34 se\_aes\_0\_key\_3

地址: 0x2000412c

se\_aes\_0\_key\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_key\_3

位	名称	权限	复位值	描述
31:0	se_aes_0_key_3	r/w	0	big endian aes key (aes-128 key LSB)

### 34.5.35 se\_aes\_0\_key\_4

地址: 0x20004130

se\_aes\_0\_key\_4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_key\_4

位	名称	权限	复位值	描述
31:0	se_aes_0_key_4	r/w	0	big endian aes key

### 34.5.36 se\_aes\_0\_key\_5

地址: 0x20004134

se\_aes\_0\_key\_5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_key\_5

位	名称	权限	复位值	描述
31:0	se_aes_0_key_5	r/w	0	big endian aes key

### 34.5.37 se\_aes\_0\_key\_6

地址: 0x20004138

**se\_aes\_0\_key\_6**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**se\_aes\_0\_key\_6**

位	名称	权限	复位值	描述
31:0	se_aes_0_key_6	r/w	0	big endian aes key

### 34.5.38 se\_aes\_0\_key\_7

地址: 0x2000413c

**se\_aes\_0\_key\_7**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**se\_aes\_0\_key\_7**

位	名称	权限	复位值	描述
31:0	se_aes_0_key_7	r/w	0	big endian aes key (aes-256 key LSB)

### 34.5.39 se\_aes\_0\_key\_sel

地址: 0x20004140

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	se_aes_0_key_sel

位	名称	权限	复位值	描述
31:2	RSVD			
1:0	se_aes_0_key_sel	r/w	0	

#### 34.5.40 se\_aes\_1\_key\_sel

地址: 0x20004144

RSVD																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

se\_aes\_1\_key\_sel

位	名称	权限	复位值	描述
31:2	RSVD			
1:0	se_aes_1_key_sel	r/w	0	

#### 34.5.41 se\_aes\_0\_endian

地址: 0x20004148

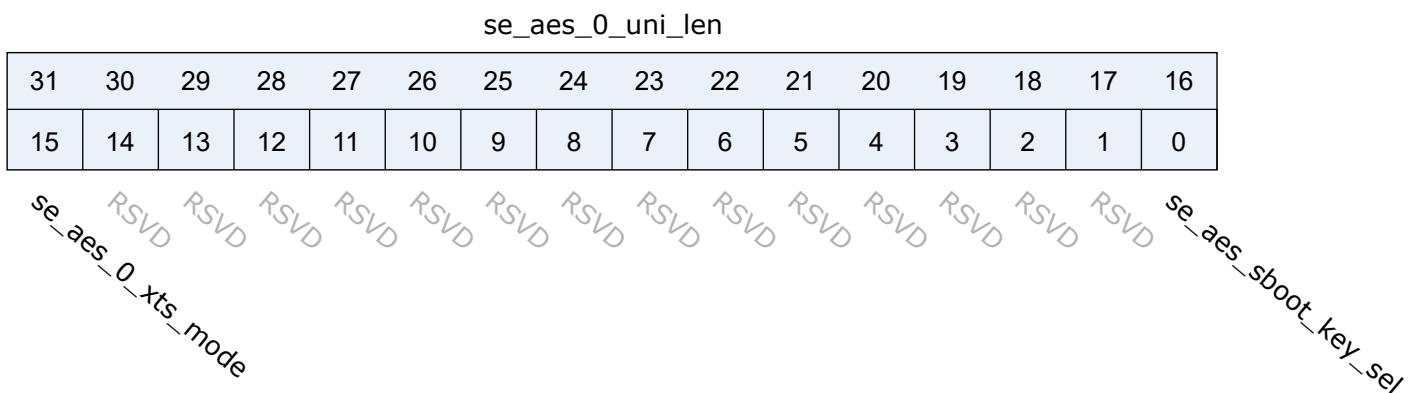
se_aes_0_ctr_len	RSVD																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

se\_aes\_0\_dout\_endian  
se\_aes\_0\_iv\_endian  
se\_aes\_0\_key\_endian  
se\_aes\_0\_twk\_endian

位	名称	权限	复位值	描述
31:30	se_aes_0_ctr_len	r/w	0	2'd0:4-byte counter, 2'd1:1-byte counter, 2'd2:2-byte counter, 2'd3:3-byte counter
29:5	RSVD			
4	se_aes_0_twk_endian	r/w	1	0:little-endian 1:big-endian, default 1 for XTS
3	se_aes_0_iv_endian	r/w	1	0:little-endian 1:big-endian
2	se_aes_0_key_endian	r/w	1	0:little-endian 1:big-endian
1	se_aes_0_din_endian	r/w	1	0:little-endian 1:big-endian
0	se_aes_0_dout_endian	r/w	1	0:little-endian 1:big-endian

#### 34.5.42 se\_aes\_sboot

地址: 0x2000414c



位	名称	权限	复位值	描述
31:16	se_aes_0_uni_len	r/w	16'd2	XTS data unit length: number of 128-bit blocks in a data unit, msg_len = N*uni_len
15	se_aes_0_xts_mode	r/w	0	0: normal XTS, 1: XTS with only one data unit
14:1	RSVD			
0	se_aes_sboot_key_sel	r/w	0	

#### 34.5.43 se\_aes\_0\_link

地址: 0x20004150

se_aes_0_lca															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_aes\_0\_lca

位	名称	权限	复位值	描述
31:0	se_aes_0_lca	r/w	0	aes link config address(word align)

#### 34.5.44 se\_aes\_0\_ctrl\_prot

地址: 0x200041fc

RSVD	RSVD	RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RSVD	se_aes_id1_en	RSVD															

位	名称	权限	复位值	描述
31:3	RSVD			
2	se_aes_id1_en	r/w	1	id1 access right
1	se_aes_id0_en	r/w	1	id0 access right
0	RSVD			

#### 34.5.45 se\_trng\_0\_ctrl\_0

地址: 0x20004200

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
se_trng_0_manual_en	se_trng_0_manual_fun_sel	RSVD	se_trng_0_int_mask	se_trng_0_int_set_1t	se_trng_0_int_clr_1t	RSVD	RSVD	RSVD	RSVD	se_trng_0_dout_clr_1t	se_trng_0_ht_error	se_trng_0_en	se_trng_0_busy	se_trng_0_trig_1t			

位	名称	权限	复位值	描述
31:16	RSVD			
15	se_trng_0_manual_en	r/w	0	1:enable manual mode
14	se_trng_0_manual_reseed	r/w	0	1:clear reseed counter to zero and get new entropy
13	se_trng_0_manual_fun_sel	r/w	0	0:go to instantiate state 1:go to generate state
12	RSVD			
11	se_trng_0_int_mask	r/w	0	
10	se_trng_0_int_set_1t	w1p	0	1:set interrupt
9	se_trng_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_trng_0_int	r	0	interrupt value
7:5	RSVD			
4	se_trng_0_ht_error	r	0	1:health test error
3	se_trng_0_dout_clr_1t	w1p	0	1:clear trng dout to zero
2	se_trng_0_en	r/w	0	0:disable 1:enable aes
1	se_trng_0_trig_1t	w1p	0	1:trigger trng engine
0	se_trng_0_busy	r	0	1:trng engine busy

#### 34.5.46 se\_trng\_0\_status

地址: 0x20004204

se\_trng\_0\_status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_status

位	名称	权限	复位值	描述
31:0	se_trng_0_status	r	32'h100020	

#### 34.5.47 se\_trng\_0\_dout\_0

地址: 0x20004208

se\_trng\_0\_dout\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_0

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_0	r	0	random value

#### 34.5.48 se\_trng\_0\_dout\_1

地址: 0x2000420c

se\_trng\_0\_dout\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_1

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_1	r	0	random value

#### 34.5.49 se\_trng\_0\_dout\_2

地址: 0x20004210

se\_trng\_0\_dout\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_2

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_2	r	0	random value

#### 34.5.50 se\_trng\_0\_dout\_3

地址: 0x20004214

se\_trng\_0\_dout\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_3

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_3	r	0	random value

### 34.5.51 se\_trng\_0\_dout\_4

地址: 0x20004218

se\_trng\_0\_dout\_4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_4

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_4	r	0	random value

### 34.5.52 se\_trng\_0\_dout\_5

地址: 0x2000421c

se\_trng\_0\_dout\_5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_5

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_5	r	0	random value

### 34.5.53 se\_trng\_0\_dout\_6

地址: 0x20004220

se\_trng\_0\_dout\_6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_6

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_6	r	0	random value

### 34.5.54 se\_trng\_0\_dout\_7

地址: 0x20004224

se\_trng\_0\_dout\_7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_dout\_7

位	名称	权限	复位值	描述
31:0	se_trng_0_dout_7	r	0	random value

### 34.5.55 se\_trng\_0\_test

地址: 0x20004228

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_ht\_alarm\_n

se\_trng\_0\_ht\_dis    se\_trng\_0\_cp\_bypass    se\_trng\_0\_test\_en  
 se\_trng\_0\_cp\_test\_en

位	名称	权限	复位值	描述
31:12	RSVD			
11:4	se_trng_0_ht_alarm_n	r/w	8'd0	health test alarm number 0:alarm if health test error >= 1 1:alarm if health test error >= 2
3	se_trng_0_ht_dis	r/w	0	1:disable health test
2	se_trng_0_cp_bypass	r/w	0	1:bypass conditional component
1	se_trng_0_cp_test_en	r/w	0	1:enable trng conditional component test mode
0	se_trng_0_test_en	r/w	0	1:enable trng test mode

### 34.5.56 se\_trng\_0\_ctrl\_1

地址: 0x2000422c

se\_trng\_0\_reseed\_n\_lsb

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_reseed\_n\_lsb

位	名称	权限	复位值	描述
31:0	se_trng_0_reseed_n_lsb	r/w	32'hffff	reload seed when number of used random value is larger than reseed_n

### 34.5.57 se\_trng\_0\_ctrl\_2

地址: 0x20004230

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_reseed\_n\_msб

位	名称	权限	复位值	描述
31:16	RSVD			
15:0	se_trng_0_reseed_n_msb	r/w	16'hff	reload seed when number of used random value is larger than reseed_n

### **34.5.58 se\_trng\_0\_ctrl\_3**

地址: 0x20004234

Method	Execution Time (s)
RSVD	~0.05
se_trng_0_ht_apt_c	~0.1
se_trng_0_rosc_en	~1.5
se_trng_0_ht_od_en	~2.5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_ht\_rct\_c

### se\_trng\_0\_cp\_ratio

位	名称	权限	复位值	描述
31	se_trng_0_rosc_en	r/w	0	trng rosc enable
30:27	RSVD			
26	se_trng_0_ht_od_en	r/w	0	health test on demand test enable
25:16	se_trng_0_ht_apt_c	r/w	10'd890	health test adaptive proportion test cut off value
15:8	se_trng_0_ht_rct_c	r/w	8'd66	health test repetition count test cut off value
7:0	se_trng_0_cp_ratio	r/w	8'd3	conditional component compression ration

#### 34.5.59 se\_trng\_0\_test\_out\_0

地址: 0x20004240

se\_trng\_0\_test\_out\_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_test\_out\_0

位	名称	权限	复位值	描述
31:0	se_trng_0_test_out_0	r	0	

#### 34.5.60 se\_trng\_0\_test\_out\_1

地址: 0x20004244

se\_trng\_0\_test\_out\_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_test\_out\_1

位	名称	权限	复位值	描述
31:0	se_trng_0_test_out_1	r	0	

### 34.5.61 se\_trng\_0\_test\_out\_2

地址: 0x20004248

se\_trng\_0\_test\_out\_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_test\_out\_2

位	名称	权限	复位值	描述
31:0	se_trng_0_test_out_2	r	0	

### 34.5.62 se\_trng\_0\_test\_out\_3

地址: 0x2000424c

se\_trng\_0\_test\_out\_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_trng\_0\_test\_out\_3

位	名称	权限	复位值	描述
31:0	se_trng_0_test_out_3	r	0	

### 34.5.63 se\_trng\_0\_ctrl\_prot

地址: 0x200042fc

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

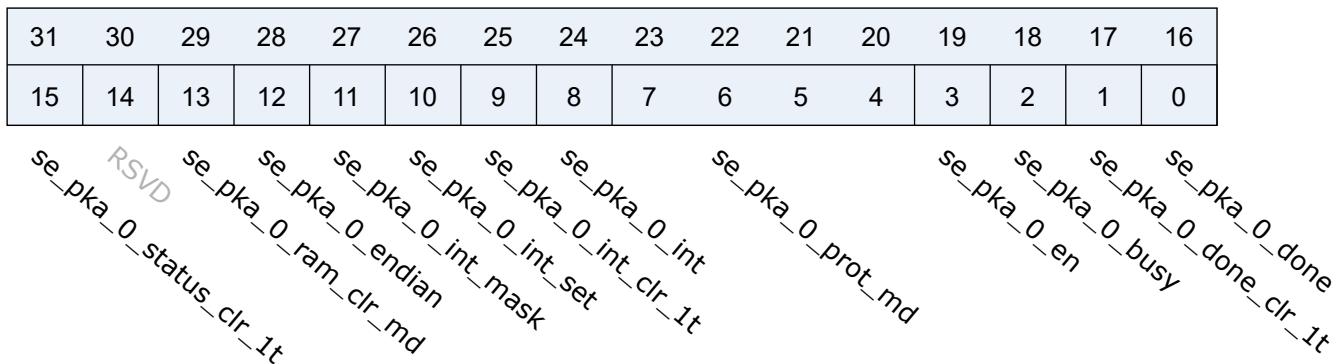
RSVD	se_trng_id0_en	se_trng_id1_en													
------	------	------	------	------	------	------	------	------	------	------	------	------	------	----------------	----------------

位	名称	权限	复位值	描述
31:3	RSVD			
2	se_trng_id1_en	r/w	1	id1 access right
1	se_trng_id0_en	r/w	1	id0 access right
0	RSVD			

### 34.5.64 se\_pka\_0\_ctrl\_0

地址: 0x20004300

se\_pka\_0\_status



位	名称	权限	复位值	描述
31:16	se_pka_0_status	r	0	[31]cmd_err, [30:26]cmd_err_idx[4:0] [25]opq_full, [24]last_opc, [23]err_cam_full, [22]err_div_by_0, [21]err_invalid_src0 [20]err_invalid_src1 [19]err_invalid_src2 [18]err_opq_overflow [17]err_unknown_opc [16]prime_fail
15	se_pka_0_status_clr_1t	w1p	0	
14	RSVD			
13	se_pka_0_ram_clr_md	r/w	0	
12	se_pka_0_endian	r/w	0	
11	se_pka_0_int_mask	r/w	0	

位	名称	权限	复位值	描述
10	se_pka_0_int_set	r/w	0	1:set interrupt
9	se_pka_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_pka_0_int	r	0	interrupt value
7:4	se_pka_0_prot_md	r/w	0	
3	se_pka_0_en	r/w	0	
2	se_pka_0_busy	r	0	
1	se_pka_0_done_clr_1t	w1p	0	
0	se_pka_0_done	r	0	

### 34.5.65 se\_pka\_0\_seed

地址: 0x2000430c

se\_pka\_0\_seed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_pka\_0\_seed

位	名称	权限	复位值	描述
31:0	se_pka_0_seed	r/w	0	

### 34.5.66 se\_pka\_0\_ctrl\_1

地址: 0x20004310

RSVD															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD

*se\_pka\_0\_hburst*  
*se\_pka\_0\_hbypass*

位	名称	权限	复位值	描述
31:4	RSVD			
3	se_pka_0_hbypass	r/w	0	
2:0	se_pka_0_hburst	r/w	3'd5	3'b000:single 3'b001:incr (undefined length) 3'b010:4-beat wrap 3'b011:4-beat incr 3'b100:8-beat wrap 3'b101:8-beat incr(default)

### 34.5.67 se\_pka\_0\_rw

地址: 0x20004340

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:0	se_pka_0_rw	r/w	0	0x340 0x35F single write for command

### 34.5.68 se\_pka\_0\_rw\_burst

地址: 0x20004360

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:0	se_pka_0_rw_burst	r/w	0	0x360 0x37F burst write for data

### 34.5.69 se\_pka\_0\_ctrl\_prot

地址: 0x2000043fc

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

RSVD    RSVD  
*se\_pka\_id1\_en*    *se\_pka\_id0\_en*

位	名称	权限	复位值	描述
31:3	RSVD			
2	se_pka_id1_en	r/w	1	id1 access right
1	se_pka_id0_en	r/w	1	id0 access right
0	RSVD			

### 34.5.70 se\_cdet\_0\_ctrl\_0

地址: 0x200004400

se_cdet_0_g_loop_min								se_cdet_0_g_loop_max							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
se_cdet_0_status															
<i>se_cdet_0_en</i> <i>se_cdet_0_error</i>															

位	名称	权限	复位值	描述
31:24	se_cdet_0_g_loop_min	r/w	8'd33	
23:16	se_cdet_0_g_loop_max	r/w	8'd100	
15:2	se_cdet_0_status	r	1	
1	se_cdet_0_error	r	0	
0	se_cdet_0_en	r/w	0	



**34.5.71 se cdet 0 ctrl 1**

地址: 0x20004404

RSVD	se_cdet_0_g_slp_n														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	名称	权限	复位值	描述
31:24	RSVD			
23:16	se_cdet_0_g_slp_n	r/w	8'd255	
15:8	se_cdet_0_t_dly_n	r/w	8'd3	
7:0	se_cdet_0_t_loop_n	r/w	8'd50	

### **34.5.72 se\_cdet\_0\_ctrl\_prot**

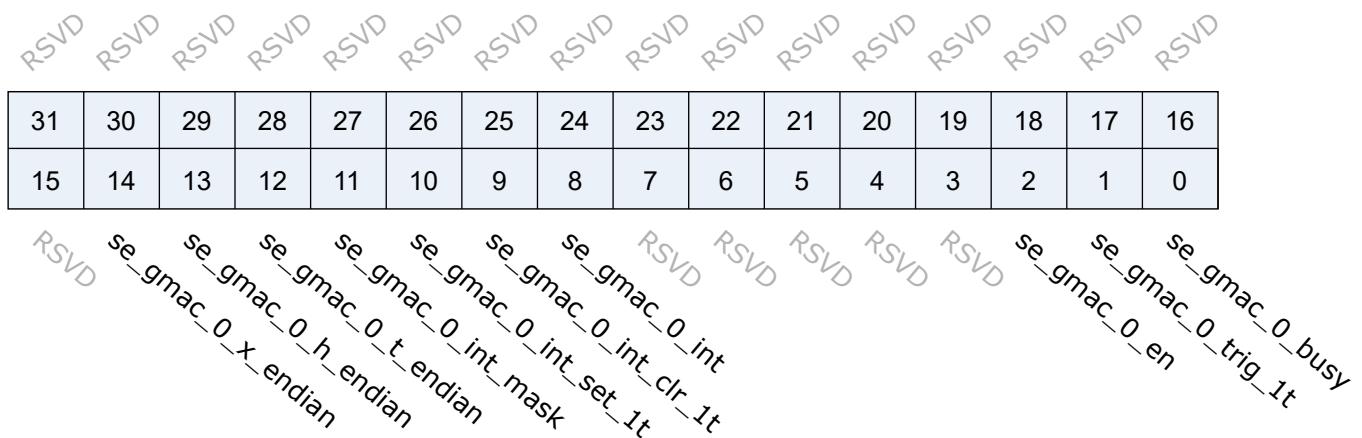
地址: 0x200044fc

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |      |      |      |      |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |      |      |      |      |

位	名称	权限	复位值	描述
31:3	RSVD			
2	se_cdet_id1_en	r/w	1	id1 access right
1	se_cdet_id0_en	r/w	1	id0 access right
0	se_cdet_prot_en	r/w	1	1:control register protection enable

### 34.5.73 se\_gmac\_0\_ctrl\_0

地址: 0x20004500



位	名称	权限	复位值	描述
31:15	RSVD			
14	se_gmac_0_x_endian	r/w	1	0:little-endian 1:big-endian
13	se_gmac_0_h_endian	r/w	1	0:little-endian 1:big-endian
12	se_gmac_0_t_endian	r/w	1	0:little-endian 1:big-endian
11	se_gmac_0_int_mask	r/w	0	1:mask interrupt
10	se_gmac_0_int_set_1t	w1p	0	1:set interrupt
9	se_gmac_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_gmac_0_int	r	0	interrupt value
7:3	RSVD			
2	se_gmac_0_en	r/w	0	0:disable 1:enable gmac
1	se_gmac_0_trig_1t	w1p	0	1:trigger gmac engine
0	se_gmac_0_busy	r	0	1:gmac engine busy

### 34.5.74 se\_gmac\_0\_lca

地址: 0x20004504

se\_gmac\_0\_lca

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_gmac\_0\_lca

位	名称	权限	复位值	描述
31:0	se_gmac_0_lca	r/w	0	gmac link config address(word align)

### 34.5.75 se\_gmac\_0\_status

地址: 0x20004508

se\_gmac\_0\_status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se\_gmac\_0\_status

位	名称	权限	复位值	描述
31:0	se_gmac_0_status	r	32'hf1000000	

### 34.5.76 se\_gmac\_0\_ctrl\_prot

地址: 0x200045fc

RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

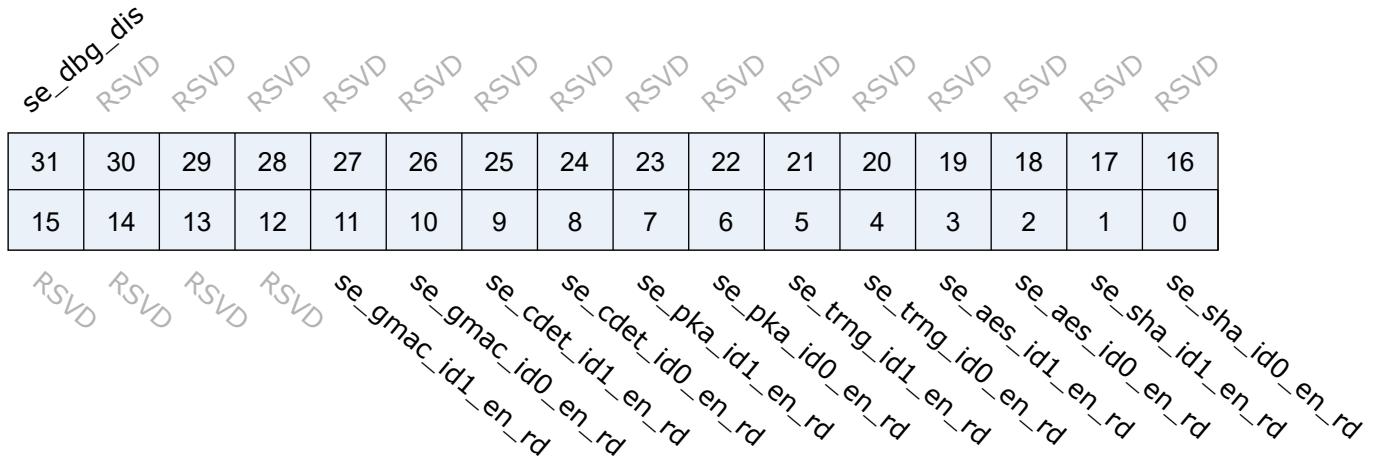
RSVD RSVD

*se\_gmac\_id1\_en* *se\_gmac\_id0\_en* RSVD

位	名称	权限	复位值	描述
31:3	RSVD			
2	se_gmac_id1_en	r/w	1	id1 access right
1	se_gmac_id0_en	r/w	1	id0 access right
0	RSVD			

### 34.5.77 se\_ctrl\_prot\_rd

地址: 0x20004f00



位	名称	权限	复位值	描述
31	se_dbg_dis	r	0	1:disable aes debug mode
30:12	RSVD			
11	se_gmac_id1_en_rd	r	1	read only status of id1 access right
10	se_gmac_id0_en_rd	r	1	read only status of id0 access right
9	se_cdet_id1_en_rd	r	1	read only status of id1 access right
8	se_cdet_id0_en_rd	r	1	read only status of id0 access right
7	se_pka_id1_en_rd	r	1	read only status of id1 access right
6	se_pka_id0_en_rd	r	1	read only status of id0 access right
5	se_trng_id1_en_rd	r	1	read only status of id1 access right
4	se_trng_id0_en_rd	r	1	read only status of id0 access right
3	se_aes_id1_en_rd	r	1	read only status of id1 access right
2	se_aes_id0_en_rd	r	1	read only status of id0 access right
1	se_sha_id1_en_rd	r	1	read only status of id1 access right
0	se_sha_id0_en_rd	r	1	read only status of id0 access right

**版本信息**

表 35.1: 文档版本修改信息

日期	版本	修改内容
2021/9/22	0.9	初版
2022/9/13	1.0	增加寄存器描述