

Sunplus Confidential for SPECTRUM Use Only



**SUNPLUS**

---

## **SPG290 PROGRAMMING GUIDE v0.5**

**2005/10/1**

SUNPLUS TECHNOLOGY CO. reserves the right to change this documentation without prior notice. Information provided by SUNPLUS TECHNOLOGY CO. is believed to be accurate and reliable. However, SUNPLUS TECHNOLOGY CO. makes no warranty for any errors which may appear in this document. Contact SUNPLUS TECHNOLOGY CO. to obtain the latest version of device specifications before placing your order. No responsibility is assumed by SUNPLUS TECHNOLOGY CO. for any infringement of patent or other rights of third parties which may result from its use. In addition, SUNPLUS products are not authorized for use as critical components in life support devices/ systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Sunplus.

SUNPLUS TECHNOLOGY CO., LTD. 19, Innovation First Road, Science-Based Industrial Park, Hsin-Chu, Taiwan, R. O. C.

% 886-3-578-6005

Æ 886-3-578-4418

H [www.sunplus.com.tw](http://www.sunplus.com.tw)

# 1 TABLE OF CONTENT

<b>1</b>	<b>TABLE OF CONTENT .....</b>	<b>2</b>
<b>2</b>	<b>REVISION HISTORY .....</b>	<b>12</b>
<b>3</b>	<b>PROCESSOR BRIEFING .....</b>	<b>13</b>
3.1	ENDIANESS.....	13
3.2	KEY FEATURES.....	13
<b>4</b>	<b>DEBUGGING IN SPG290A .....</b>	<b>14</b>
4.1	OVERVIEW .....	14
4.2	SJTAG MODULE.....	14
4.3	RS232.....	15
4.4	REFERENCE.....	15
<b>5</b>	<b>S+CORE IDE.....</b>	<b>16</b>
5.1	INSTALLATION S+CORE IDE .....	16
5.2	EXTERNAL DEVICE FOR SIMULATOR .....	16
<b>6</b>	<b>SCENEDIRECTOR - GRAPHIC TOOL .....</b>	<b>17</b>
6.1	FEATURES.....	17
6.2	INSTALLING SCENEDIRECTOR .....	17
6.2.1	System Requirement.....	17
6.2.2	Installing.....	18
<b>7</b>	<b>SYSTEM OVERVIEW .....</b>	<b>19</b>
7.1	GENERAL DESCRIPTION .....	19
<b>8</b>	<b>MEMORY INTERFACE UNIT - MIU.....</b>	<b>20</b>
8.1	MEMORY MAPPING .....	21
8.2	SYSTEM REGISTERS : EACH MODULE USED 16K BYTES RANGE.....	21
8.3	CONTROL REGISTERS.....	22
8.3.1	CSI frame buffer start address .....	22
8.3.2	PPU frame buffer start address .....	22
8.3.3	P_SPU_START_ADR(0x88070054).....	22
8.3.4	P_SDRAM_POWER_DOWN (0x8807005C).....	23
8.3.5	P_MIU1_SDRAM_SETTING(0x88070060) .....	23
8.3.6	P_MIU_STATUS(0x8807006C) .....	24
8.3.7	MP4 RAW Image frame buffer.....	24
8.3.8	MP4 write frame buffer.....	24
8.3.9	MP4 VLC buffer.....	24
8.3.10	P_MP4_FRAME_BUF_HSIZE(0x88070090) .....	24
8.3.11	P_SDRAM_SETTING2(0x88070094).....	25

8.4	ARBITER.....	25
<b>9</b>	<b>2D PICTURE PROCESS UNIT - PPU .....</b>	<b>26</b>
9.1	ORDER OF BYTES.....	27
9.2	COLOR MODE.....	27
9.2.1	<i>Color modes of SPRITE</i> .....	27
9.2.2	<i>Color modes of TEXT</i> .....	27
9.3	PALETTE MEMORY .....	28
9.3.1	<i>Palette Bank</i> .....	28
9.3.2	<i>Pixel format of Palette</i> .....	28
9.4	ATTRIBUTES .....	29
9.4.1	<i>Palette</i> .....	29
9.4.2	<i>Depth</i> .....	29
9.4.3	<i>Character Flip</i> .....	31
9.4.4	<i>Character Size</i> .....	32
9.4.5	<i>Character Data</i> .....	33
9.5	TEXT SCREEN.....	34
9.5.1	<i>Character Mode</i> .....	34
9.5.2	<i>Bitmap Mode</i> .....	34
9.6	TEXT COORDINATE.....	35
9.7	SPRITE COORDINATE.....	38
9.8	TEXT HORIZONTAL MOVEMENT CONTROL.....	39
9.9	TEXT COMPRESSION/EXTENSION CONTROL.....	41
9.9.1	<i>Vertical compression</i> .....	41
9.9.2	<i>Horizontal compression</i> .....	44
9.9.3	<i>Horizontal and Vertical Compression</i> .....	46
9.9.4	<i>Horizontal/Vertical Compression and Movement</i> .....	46
9.10	SPRITE MEMORY .....	47
9.11	TEXT MEMORY.....	47
9.12	WALLPAPER MODE.....	49
9.13	BLENDING EFFECT .....	50
9.14	32768/65536 HIGH COLOR MODE .....	50
9.15	CHARACTER ATTRIBUTES.....	51
9.16	BITMAP MODE ATTRIBUTES .....	52
9.17	ADDRESSING MODE .....	53
9.18	PPU API SERVICES .....	54
9.19	PPU REGISTER SET .....	63
<b>10</b>	<b>BITBLT DMA WITH ALPHA BLENDING OPERATION .....</b>	<b>77</b>

10.1	FEATURES.....	77
10.2	ARCHITECTURE.....	78
10.3	CONTROL REGISTERS.....	79
10.3.1	<i>P_BLNDMA_SRC_A_ADDR(R/W)(0x880D0000): DMA SourceA Starting Address</i> .....	79
10.3.2	<i>P_BLNDMA_SRC_B_ADDR(R/W)(0x880D0004): DMA SourceB Starting Address</i> .....	79
10.3.3	<i>P_BLNDMA_DEST_ADDR(R/W)(0x880D0008): DMA Destination Starting Address</i> .....	79
10.3.4	<i>P_BLNDMA_WIDTH_HEIGHT(R/W)(0x880D000C): DMA Transfer Width</i> .....	79
10.3.5	<i>P_BLNDMA_FILL_PAT(R/W)(0x880D0010): DMA Fill Pattern</i> .....	79
10.3.6	<i>P_BLNDMA_CONTROL_1(R/W)(0x880D0014): DMA Operation Mode</i> .....	79
10.3.7	<i>P_BLNDMA_IRQ_CONTROL(R/W)(0x880D0018): DMA Fill Pattern</i> .....	80
10.3.8	<i>P_BLNDMA_BLEND_FACTOR(R/W)(0x880D001C):</i> .....	80
10.3.9	<i>P_BLNDMA_TRANSPARENT(R/W)(0x880D0020):</i> .....	80
10.3.10	<i>P_BLNDMA_ADDR_MODE(R/W)(0x880D0024):</i> .....	80
10.3.11	<i>P_BLNDMA_CONTROL_2(R/W)(0x880D0028):</i> .....	80
10.3.12	<i>P_BLNDMA_ABASE_ADDR(R/W)(0x880D0030):</i> .....	81
10.3.13	<i>P_BLNDMA_AOFFSET_XY(R/W)(0x880D0034):</i> .....	81
10.3.14	<i>P_BLNDMA_A_BG(R/W)(0x880D0038):</i> .....	81
10.3.15	<i>P_BLNDMA_BBASE_ADDR(R/W)(0x880D0040):</i> .....	81
10.3.16	<i>P_BLNDMA_BOFFSET_XY(R/W)(0x880D0044):</i> .....	81
10.3.17	<i>P_BLNDMA_B_BG(R/W)(0x880D0048):</i> .....	81
10.3.18	<i>P_BLNDMA_DBASE_ADDR(R/W)(0x880D0050):</i> .....	82
10.3.19	<i>P_BLNDMA_DOFFSET_XY(R/W)(0x880D0054):</i> .....	82
10.3.20	<i>P_BLNDMA_D_BG(R/W)(0x880D0058):</i> .....	82
10.4	BLNDMA API SERVICES.....	83
<b>11</b>	<b>DISPLAY UNIT - TVE .....</b>	<b>108</b>
11.1	DESCRIPTION .....	108
11.2	TV SYSTEM AND SCREEN MODE .....	108
11.3	FRAME BUFFER .....	108
11.4	DISPLAY .....	108
11.5	TVE REGISTERS.....	108
11.5.1	<i>P_TV_Control</i> .....	108
11.5.2	<i>TV Saturation &amp; Hue Adjust</i> .....	109
11.6	TV ENCODE API SERVICES .....	110
<b>12</b>	<b>SOUND PROCESS UNIT - SPU .....</b>	<b>112</b>
12.1	GENERAL DESCRIPTION .....	112
12.2	FEATURE .....	112
12.3	BLOCK DIAGRAM.....	113

12.4	INTERNAL MEMORY MAPPING .....	114
12.5	GAIN CONTROL.....	115
12.6	FUNDAMENTALS.....	116
12.7	CONTROL REGISTER.....	118
12.8	INTERNAL SRAM.....	133
12.9	S/W PROGRAMMING FLOW EXAMPLE .....	147
<b>13</b>	<b>MP4 CODEC.....</b>	<b>149</b>
<b>14</b>	<b>CMOS SENSOR APPLICATION.....</b>	<b>150</b>
14.1	FRAME BUFFER .....	150
14.2	BUFFER CONTROL .....	150
14.3	BUS INTERFACE.....	151
14.4	CONTROL REGISTERS.....	151
14.5	TIME GENERATOR .....	151
14.5.1	<i>P_TG_CR (0x08000000): Time Generator Control.....</i>	<i>152</i>
14.5.2	<i>P_TG_LSTART(0x08000004): .....</i>	<i>153</i>
14.5.3	<i>P_TG_START(0x08000008): .....</i>	<i>154</i>
14.5.4	<i>P_TG_END(0x0800000C):.....</i>	<i>154</i>
14.5.5	<i>P_TG_BLACK(0x0800000C): .....</i>	<i>154</i>
14.5.6	<i>P_TG_BSUPPER(0x08000014): .....</i>	<i>155</i>
14.5.7	<i>P_TG_BSLOWER(0x08000018):.....</i>	<i>155</i>
14.5.8	<i>P_TG_TRANSP(0x0800001C): .....</i>	<i>156</i>
14.5.9	<i>P_TG_FBSADDR1(0x08000020): Frame Buffer 1's start address.....</i>	<i>156</i>
14.5.10	<i>P_TG_FBSADDR2(0x08000024): Frame Buffer 2's start address.....</i>	<i>156</i>
14.5.11	<i>P_TG_FBSADDR3(0x08000028): Frame Buffer 3's start address.....</i>	<i>156</i>
14.5.12	<i>P_TG_CAP(0x0800002C): Frame Buffer 3's start address .....</i>	<i>157</i>
14.6	MOTION DETECT CONTROL .....	157
14.6.1	<i>P_MD_CR(0x88000030): Motion Detect control register. ....</i>	<i>157</i>
14.6.2	<i>P_MD_SADDR(0x88000034): .....</i>	<i>158</i>
14.6.3	<i>P_MD_POS(0x88000038):.....</i>	<i>158</i>
14.6.4	<i>P_MD_SADDR1(0x8800003C):.....</i>	<i>159</i>
14.6.5	<i>P_MD_CTABLE0(0x88000040):.....</i>	<i>159</i>
14.6.6	<i>P_MD_CTABLE1(0x88000044):.....</i>	<i>159</i>
14.6.7	<i>P_MD_CTABLE2(0x88000048):.....</i>	<i>159</i>
14.6.8	<i>P_MD_CTABLE3(0x8800004C): .....</i>	<i>159</i>
14.6.9	<i>P_MD_REG1(0x88000050): .....</i>	<i>159</i>
14.6.10	<i>P_MD_REG2(0x88000054):.....</i>	<i>160</i>
14.6.11	<i>P_MD_REG3(0x88000058): .....</i>	<i>160</i>



14.6.12	<i>P_MD_TH(0x8800005C):</i> .....	160
14.6.13	<i>P_MD_RGB(0x88000074):</i> .....	160
14.7	INTERRUPT.....	161
14.7.1	<i>P_CSI_IRQEN(0x88000078)</i> .....	161
14.7.2	<i>P_CSI_IRQSTS(0x8800007C)</i> .....	161
14.8	COLOR MODE.....	163
14.8.1	<i>P_CSI_Y2R_A1(0x880000E8)</i> .....	163
14.8.2	<i>P_CSI_Y2R_A2(0x880000EC)</i> .....	163
14.8.3	<i>P_CSI_Y2R_A3(0x880000F0)</i> .....	163
14.8.4	<i>P_CSI_R2Y_A1(0x880000E8)</i> .....	163
14.8.5	<i>P_CSI_R2Y_A2(0x880000EC)</i> .....	163
14.8.6	<i>P_CSI_R2Y_A3(0x880000F0)</i> .....	164
14.9	MOTION DETECTION.....	164
14.9.1	<i>Recognition Sample Type</i> .....	164
14.9.2	<i>Recognition Reference Filed / Frame</i> .....	165
15	CCIR601/656 VIDEO INPUT.....	167
16	USB 1.1.....	168
16.1.1	<i>USB Device</i> .....	169
16.1.2	<i>USB Mini Host</i> .....	169
16.1.3	<i>Control Register</i> .....	169
17	SD CARD CONTROLLER.....	199
17.1	FEATURES.....	199
17.2	BLOCK DIAGRAM.....	200
17.3	CONTROL REGISTER.....	200
17.3.1	<i>P_SD_DATATx (R/W)(0x88180000) : SD Card Data Transmit register</i> .....	200
17.3.2	<i>P_SD_DATARx(R) (0x88180004): SD Card Data Receive register</i> .....	200
17.3.3	<i>P_SD_COMMAND(R/W)(0x88180008): SD Card Command register</i> .....	201
17.3.4	<i>P_SD_ARGUMENT(R/W)(0x8818000C): SD Card argument register</i> .....	202
17.3.5	<i>P_SD_STATUS (R)(0x88180014): SD Card Controller status register</i> .....	202
17.3.6	<i>P_SD_RESPONSE (R) (0x88180010): SD Card Response register</i> .....	203
17.3.7	<i>P_SD_CONTROL(R/W)(0x88180018): SD Card Control Register</i> .....	204
17.3.8	<i>13.10 P_SD_INTEN(R/W) (0x8818001C): SD Card Interrupt Register</i> .....	204
18	UNIVERSAL FILE SYSTEM – UFAT LIBRARY.....	206
18.1	STRUCTURE.....	206
18.2	API FUNCTIONS.....	207
18.2.1	<i>open</i> .....	207
18.2.2	<i>close</i> .....	207

18.2.3	read.....	208
18.2.4	write.....	208
<b>19</b>	<b>CD SERVO CONTROL .....</b>	<b>209</b>
19.1	ARCHITECTURE.....	209
19.2	DATA BUFFER IN DRAM .....	209
19.3	BUS INTERFACE.....	210
19.4	CONTROL REGISTERS.....	210
19.5	INTERRUPT .....	210
<b>20</b>	<b>LCD TIMING CONTROL .....</b>	<b>211</b>
20.1	FRAME BUFFER .....	211
20.2	BUFFER CONTROL .....	211
20.3	BUS INTERFACE.....	211
20.4	CLOCK INPUT PORTS .....	212
20.5	CONTROL REGISTERS.....	212
20.6	INTERRUPT .....	212
20.7	SUPPORTED LCD PANELS .....	212
<b>21</b>	<b>LIGHT GUN APPLICATION.....</b>	<b>213</b>
21.1	INTRODUCTION .....	213
21.2	PRINCIPLE OF LIGHTGUN.....	214
21.3	TEST ENVIRONMENT .....	214
21.4	LIGHTGUN CONTROL .....	215
21.5	LIGHTGUN FUNCTION.....	215
21.6	EXAMPLE PROGRAM .....	215
<b>22</b>	<b>BUFFER CONTROL - BUFCTL .....</b>	<b>218</b>
22.1	CONTROL REGISTERS.....	218
22.1.1	<i>P_PTR_SETTING(R/W)(0x88090004): Control type .....</i>	<i>218</i>
22.1.2	<i>P_PPU1_Buf_PTR_REG(R/W)(0x8809000C): Software buffer point for Text1.....</i>	<i>219</i>
22.1.3	<i>P_PPU2_Buf_PTR_REG(R/W)(0x88090010): Software buffer point for Text2 .....</i>	<i>219</i>
22.1.4	<i>P_PPU3_Buf_PTR_REG(R/W)(0x88090014): Software buffer point for Text3 .....</i>	<i>219</i>
22.1.5	<i>P_TVE_Buf_PTR_REG(R/W)(0x88090020): Software buffer point for TVE .....</i>	<i>219</i>
22.1.6	<i>P_P2T_SETTING(R/W)(0x8809003C): PPU to TVE buffer control register .....</i>	<i>220</i>
22.1.7	<i>P_BUFCTL_SETTING(R/W)(0x8809004C): Buffer Control Register .....</i>	<i>220</i>
<b>23</b>	<b>UART.....</b>	<b>221</b>
23.1	BLOCK DIAGRAM.....	221
23.2	CONTROL REGISTERS .....	222
23.2.1	<i>P_UART_Data(R/W)(0x88150000): UART Data Tx/Rx Register. ....</i>	<i>222</i>
23.2.2	<i>P_UART_ERR(R/W)(0x88150004): UART Tx &amp; Rx Error Flag Register.....</i>	<i>222</i>



23.2.3	<i>P_UART_Ctrl(R/W)(0x88150008):UART Control Register</i> .....	223
23.2.4	<i>P_UART_BaudRate(R/W)(0x8815000C):UART Baud Rate Setup Register</i> .....	224
23.2.5	<i>P_UART_Status(R/W)(0x88150010):UART Status Register</i> .....	224
<b>24</b>	<b>SERIAL INTERFACE I/O - SIO</b> .....	<b>226</b>
24.1	GENERAL DESCRIPTION .....	226
24.2	SIO PROTOCOL .....	226
24.3	SIO FEATURE .....	226
24.4	SIO CONTROL REGISTERS .....	227
24.4.1	<i>P_SIO_Control(0x88120000)</i> .....	227
24.4.2	<i>P_SIO_AutoTx (0x88120004): Automatic transmit word number</i> .....	228
24.4.3	<i>P_SIO_Addr(0x88120008): SIO Start Address Register</i> .....	229
24.4.4	<i>P_SIO_DATA(0x8812000C): SIO Data Register</i> .....	229
24.5	SIO INTERRUPT PROCESSING .....	230
24.6	SIO USAGE PROCEDURE .....	231
<b>25</b>	<b>SERIAL PERIPHERAL INTERFACE - SPI</b> .....	<b>234</b>
25.1	GENERAL DESCRIPTION .....	234
25.2	FEATURES .....	234
25.3	ARCHITECTURE .....	235
25.4	CONTROL REGISTERS .....	235
25.4.1	<i>P_SPI_Control(0x88110000)</i> .....	235
25.4.2	<i>P_SPI_TX_Status (0x88110004)</i> .....	236
25.4.3	<i>P_SPI_TX_DATA (0x88110008)</i> .....	236
25.4.4	<i>P_SPI_RX_Status (0x8811000C)</i> .....	236
25.4.5	<i>P_SPI_RX_DATA (0x88110008)</i> .....	237
25.4.6	<i>P_SPI_RX_Status (0x8811000C)</i> .....	237
<b>26</b>	<b>PERIPHERAL INTERRUPT CONTROLLER</b> .....	<b>241</b>
26.1	INTRODUCTION .....	241
26.2	FEATURE .....	241
26.3	CONTROL REGISTER .....	242
26.3.1	<i>P_INTPND( R)(0x880A0000): INTPND Register</i> .....	242
26.3.2	<i>P_INTPND_H( R)(0x880A0004):</i> .....	242
26.3.3	<i>P_I_PMST(R/W)(0x880A0008)</i> .....	242
26.3.4	<i>P_I_GPUSWI(R/W)(0x880A000C)</i> .....	243
26.3.5	<i>IRQ Priority Register</i> .....	243
26.4	VECADDR .....	243
26.5	INTERRUPT SOURCES .....	244
<b>27</b>	<b>TIMER</b> .....	<b>245</b>



27.1	FEATURES.....	245
27.2	ARCHITECTURE.....	245
27.3	CONTROL REGISTERS.....	247
27.3.1	P_TimerXCtrl(0x8816X000).....	247
27.3.2	P_CPP_CTRL(0x8816X004).....	248
27.3.3	P_Preload_REGS(0x8816X008).....	248
27.3.4	P_CPP_REGS(0x8816X00C).....	248
27.3.5	P_CPP_UPCOUNT(0x8816X010).....	248
28	RTC.....	249
29	wDOG.....	250
30	SLEEP/WAKEUP.....	251
31	SOFTWARE CONFIG - SFTCFG.....	252
31.1	FUNCTION DESCRIPTION.....	252
31.2	ARCHITECTURE.....	252
31.3	CONTROL REGISTERS.....	253
31.3.1	P_GPIO_DEVICE_SET(0x88200000).....	253
31.3.2	P_GPIO_FUNC_SET(0x88200004).....	253
31.3.3	P_GPIO_DRAM_SET(0x88200008).....	254
31.3.4	P_GPIO_OUTPUT(0x8820000C).....	254
31.3.5	P_GPIO_TFT_PORT(0x88200010).....	254
31.3.6	P_GPIO_TFT_OE(0x88200014).....	254
31.3.7	P_GPIO_TFT_PU(0x88200018).....	254
31.3.8	P_GPIO_TFT_PD(0x8820001C).....	255
31.3.9	P_GPIO_CSI_PORT(0x88200020).....	255
31.3.10	P_GPIO_CSI_PULL(0x88200024).....	255
31.3.11	P_GPIO_NFLASH_PORT(0x88200028).....	255
31.3.12	P_GPIO_NFLASH_PULL(0x8820002C).....	255
31.3.13	P_GPIO_JTAG_PORT(0x88200030).....	256
31.3.14	P_GPIO_GLOBAL_PORT(0x88200034).....	256
31.3.15	P_GPIO_USB_PORT(0x88200038).....	256
31.3.16	P_GPIO_UART_PORT(0x8820003C).....	256
31.3.17	P_GPIO_I2C_PORT(0x88200040).....	257
31.3.18	P_GPIO_ADC_PORT(0x88200044).....	257
31.3.19	P_GPIO_DRAM_PORT(0x88200048).....	257
31.3.20	P_GPIO_ADC_AEN(0x8820004C).....	258
32	JPEG.....	259
33	APBDMA.....	260

33.1	ARCHITECTURE.....	260
33.2	APB BRIDGE.....	260
33.3	DMA CONTROLLER .....	260
33.4	CONTROL REGISTERS.....	261
33.4.1	<i>P_CH_BUSY_STS(0x88080000)</i> .....	261
33.4.2	<i>P_CH_IRQ_STS(0x88080004)</i> .....	261
33.4.3	<i>Buffer A Start Address</i> .....	262
33.4.4	<i>Buffer A End Address</i> .....	262
33.4.5	<i>APB Module Access Port Start Address</i> .....	262
33.4.6	<i>Buffer B Start Address</i> .....	262
33.4.7	<i>Buffer B End Address</i> .....	263
33.4.8	<i>P_CH1_SETTING (0x8808006C)</i> .....	263
33.4.9	<i>P_CH2_SETTING (0x88080070)</i> .....	264
33.4.10	<i>P_CH3_SETTING (0x88080074)</i> .....	264
33.4.11	<i>P_CH4_SETTING (0x88080078)</i> .....	265
33.4.12	<i>P_CH_SW_RST (0x8808007C)</i> .....	266
33.5	DMA END.....	266
<b>34</b>	<b>NAND-TYPE FLASH.....</b>	<b>267</b>
34.1	INTRODUCTION.....	267
34.2	FEATURE .....	267
34.3	BLOCK DIAGRAM .....	267
34.4	INTERFACE SIGNAL.....	267
34.4.1	<i>APB Bus Signals</i> .....	267
34.4.2	<i>Flash Bus Signals</i> .....	268
34.4.3	<i>Other Signals</i> .....	268
34.5	CONTROL REGISTERS.....	268
34.5.1	<i>P_FL_CR(R/W)(0x8800F000): Control Register</i> .....	268
34.5.2	<i>P_FL_CLE(R/W)(0x8800F004): Command Latch enable register</i> .....	270
34.5.3	<i>P_FL_ALE(R/W)(0x8800F008): Address latch enable register</i> .....	270
34.5.4	<i>P_FL_WD(R/W)(0x8800F00C): Write data register</i> .....	270
34.5.5	<i>P_FL_RD(R/W)(0x8800F010): Read data register</i> .....	270
34.5.6	<i>P_FL_INTEN(R/W)(0x8800F014): Interrupt enable register</i> .....	271
34.5.7	<i>P_FL_INTSTS(R/C)(0x8800F018): Interrupt status register</i> .....	271
34.5.8	<i>P_FL_TRUELP(R/W)(0x8800F01C): True line parity register</i> .....	272
34.5.9	<i>P_FL_TRUECP(R/W)(0x8800F020): True column parity register</i> .....	273
34.5.10	<i>P_FL_CALLP(R)(0x8800F024): Calculate line a parity register</i> .....	273
34.5.11	<i>P_FL_CALCP(R)(0x8800F028): Calculate column parity register</i> .....	273



34.5.12	<i>P_FL_ECCSTS(R)(0x8800F02C): ECC status register.....</i>	<i>274</i>
34.5.13	<i>P_FL_CALECC(W)(0x8800F030): ECC control register .....</i>	<i>274</i>
<b>35</b>	<b>INTER-INTEGRATED CIRCUIT - I2C .....</b>	<b>275</b>
35.1	INITIALIZATION .....	275
35.2	CONTROL REGISTERS .....	276
35.2.1	<i>P_I2C_CR(R/W)(0x88130020): I2C configuration register. ....</i>	<i>276</i>
35.2.2	<i>P_I2C_INTR(R/W)(0x88130024): I2C interrupt register. ....</i>	<i>277</i>
35.2.3	<i>P_I2C_CVR(R/W)(0x88130028): I2C clock setting register.....</i>	<i>277</i>
35.2.4	<i>P_I2C_ID(R/W)(0x8813002C): I2C ID register. ....</i>	<i>277</i>
35.2.5	<i>P_I2C_ADDR(R/W)(0x88130030): I2C port address register.....</i>	<i>277</i>
35.2.6	<i>P_I2C_WDATA(R/W)(0x88130034): I2C write data register. ....</i>	<i>277</i>
35.2.7	<i>P_I2C_RDATA(R/W)(0x88130038): I2C read data register. ....</i>	<i>277</i>
<b>36</b>	<b>INTERACTIVE INFORMATION SERVICES - I2S.....</b>	<b>278</b>
36.1	FEATURES.....	278
36.2	ARCHITECTURE.....	278
36.3	CONTROL REGISTERS .....	278
36.3.1	<i>P_I2S_CR(R/W)(0x88140000): I2S control.....</i>	<i>278</i>
36.3.2	<i>P_I2S_IRQ_STS(R/W)(0x88140004): I2S IRQ Status .....</i>	<i>279</i>
36.3.3	<i>P_I2S_FIFO(R/W)(0x88140008): I2S FIFO Config .....</i>	<i>279</i>
36.3.4	<i>P_I2S_RDATA(R/W)(0x8814000C): I2S receive data .....</i>	<i>280</i>
<b>37</b>	<b>LOCAL DATA MEMORY - LDM.....</b>	<b>281</b>
37.1	ARCHITECTURE.....	281
37.2	LDMDMA CONTROLLER.....	281
37.3	CONTROL REGISTERS.....	282
37.3.1	<i>P_DMA_CTRL(0x880C0000).....</i>	<i>282</i>
37.3.2	<i>P_DMA_STATUS(0x880C0004).....</i>	<i>282</i>
37.3.3	<i>P_MIU_START_ADDR(0x880C0008).....</i>	<i>282</i>
37.3.4	<i>P_MIU_END_ADDR(0x880C000C).....</i>	<i>282</i>
37.3.5	<i>P_LDM_START_ADDR(0x880C0010).....</i>	<i>282</i>
37.3.6	<i>P_LDM_END_ADDR(0x880C0014).....</i>	<i>283</i>
<b>38</b>	<b>ADC .....</b>	<b>284</b>
38.1	FEATURES.....	284
38.2	ARCHITECTURE.....	284
38.3	CONTROL REGISTERS .....	285
38.3.1	<i>P_ADC_SETUP(0x881A0000): ADC Setup .....</i>	<i>285</i>
38.3.2	<i>P_MIC_GAIN(0x881A0004): MIC GAIN.....</i>	<i>286</i>
38.3.3	<i>P_ADC_CLOCK_SET(0x881A0008): ADC Clock Setup .....</i>	<i>286</i>

38.3.4 P\_ADC\_HOLD\_SETUP(0x881A000C): Sample Hold Setup ..... 286

---

---

## 2 REVISION HISTORY

---

---

Revision	Date	By	Remark
0.1	05/30/2005	Wang shu	First edition
0.2	08/25/2005	Jackie Chang Joe Chen	Second edition
0.3/0.4	08/25/2005	Jackie Chang	For internal development review only
0.5	10/01/2005	Jackie Chang Joe Chen	All chapters are base on fixed design specification

### 3 PROCESSOR BRIEFING

**S+Core** is a 32-bit RISC with Proprietary **Instruction Set Architecture (ISA)** designed and invented by Sunplus. The ISA has 32/16 bits hybrid instruction mode and parallel conditional execution (patent pending) for high code density, high performance and versatile applications. The micro-architecture includes AMBA bus for SoC integration, co-processor, custom engine interface for more flexible function integration, and also SJTAG interface for more convenient ICE environment. The data access in SPG290 system is in **little-endian** architecture.

#### 3.1 Endianess

All data access in SPG290 32 bits system are **Little Endian**. The most significant byte in a WORD, 32 bits data, is at the last place of a 4 bytes string.

For data read operations, a simple example is show below for word(32 bits), half-word(16-bits), and byte(8-bits) access:

<b>Memory Address</b>	0x00	0x01	0x02	0x03	<b>Read word</b> 0x00	0x78563412	
<b>Data Content</b>	0x12	0x34	0x56	0x78	<b>Read half word</b> 0x00m 0x03	0x3412	0x7856
					<b>Read byte</b> 0x00~0x03	0x12	0x34 0x56 0x78

#### 3.2 Key Features

The features of S+Core are listed as following:

- I 32/16-bits Hybrid Execution Mode
- I Parallel Conditional Execution(patent pending)
- I Capability for Further Software Security Design(Patent pending)
- I A Harvard Architecture(I-Cache/ D-Cache) solution
- I MMU (Fixed mode and Full-function mode)
- I Compliance to the AMBA Specification (Rev 2.0) for easy integration into SOC implementation
- I Vectored Interrupt
- I SJTAG (Sunplus JTAG)

---

---

## 4 DEBUGGING IN SPG290A

---

---

### 4.1 Overview

The debug solution in processor includes following features:

**I Off-chip debug memory access**

SJTAG allows processor in the debug mode access the instructions or data that are not physically on the chip. Memory access to the special segment will be transferred to JTAG protocol and then sent to the debug probe that controlled by debug host PC. Debug probe will handle the memory access by redirect the access to its local memory. Then, the accessed data will feedback to processor via JTAG again.

**I Hardware breakpoint**

Two types of hardware breakpoint are included which can be configured to cause debug exception on:

- 1) Instruction fetch of a specified address(Breakpoint)
- 2) Data fetch of a specified address and the access data value(Watchpoint)

**I Software breakpoint instruction**

Two additional instructions are added to achieve system debug: Software Debug Breakpoint(SDBBP) and Debug Exception Return(DRET).

**I Single step execution**

A dedicate single step exception in processor is provided for single step debugging.

**I Debug interrupt**

A debug interrupt is provided to force processor enter debug mode at any time.

**I DMA Access**

A DMA channel that controlled by SJTAG directly access system bus through BIU. This feature is very useful when user need to download code to system memory.

### 4.2 SJTAG Module

The SJTAG module includes:

**I The hardware point unit will assert the breakpoint exception request to core.**

**I A memory controller handles the memory request from core if the address is located in debug segment. If the access is in debug register segment, it will directly access the debug registers such as breakpoint unit registers or debug control register. If the access is in debug memory segment, it will pass the access to debug probe through JTAG interface.**

- I A JTAG controller handles the communication between probe and SJTAG module.

#### **4.3 RS232**

The SPG290A still provides a RS232 for debugging purposes. The user should be acquainted with the I/O function printf(). In addition, during the development phase, a simple command monitor is implemented to allow user to issue commands via the RS232. For example, the user can issue a dump command to dump memory content.

#### **4.4 Reference**

- I **S+Core 7 Processor Core Technical Reference Manual( for Software Use) V2.1 – Feb. 01, 2005, by Sunplus Technology Co., Ltd.**
- I **S+Core IDE User Manual, by Sunplus Technology Co., Ltd.**

---

---

## 5 S+CORE IDE

---

---

S+Core IDE, a 32-bit powerful Integrated Development Environment(IDE) for developing application in C or assembler for S+Core series CPU, which owns fully integrated Windows development tool.

### 5.1 Installation S+Core IDE

The S+Core IDE can be run on Windows98<sup>®</sup>, Windows2000<sup>®</sup> and WindowsXP<sup>®</sup>. Follow the on-screen instruction and S+Core IDE will be installed on your computer. The following two points must be considered:

1. You must have administrator privilege when you install S+Core IDE on Win2000/XP.
2. The printer-port mode of "SPP" must be set in BIOS setting, and hardware address must be set to 378H as well.

### 5.2 External Device for Simulator

In Simulator mode, the S+Core IDE provides two external devices for easy debugging your program.

#### I LCD

LCD displaying can be show during running after LCD is enabled from [Project]->[Setting]->[Simulator]-[Peripheral equipments].

NOTES:

1. LCD displaying can be used in simulator mode only.
2. Default LCD is off

#### I UART

UART printf can be enabled during running after UART is enabled from [Project] [Setting] [Simulator] [Peripheral equipments]. For more information about UART printf usage, please refer to example "UART printf" provided in installation kit.

NOTES:

1. UART printf can be used in simulator mode only.
2. Default UART is off



---

---

## 6 SCENEDIRECTOR - GRAPHIC TOOL

---

---

SceneDirector™, a powerful tool authored by Sunplus Tech Co., Ltd mainly for developing games, offers effective ways to import and edit pictures, to setup and view an animation (especially, to view an animation on an ICE), to export animation data to PC, to manage [cells](#) (small picture), and to edit palettes. With the user-friendly interface and flexible editing modes, the SceneDirector is capable of helping you to work easily and efficiently.

### 6.1 Features

- I Single-project architecture for organizing Textures, Bitmaps, JPEGS, Layers, and Sprites(frames)
- I Supports the following formats: \*.bmp \*.jpg \*.pcx \*.gif
- I Provides two modes for importing pictures: sliced(into cells), unsliced.
- I Two cell libraries: Scene Library and System Library
- I Two editable, exportable, independent palette bars, up to 512 colors.
- I RGB555/RGB565 mode for Bitmaps
- I Powerful edit function on textures, bitmaps, layers, sprites, and frames.
- I Flexible animation setup and preview function with one background(Layers item) and several sprites.
- I Up to 3 layers for a background.
- I Various Download function: download image(texture, frame, bitmap, JPEG)/ animation, download to PC/ICE
- I Two export modes: ASM, BIN.
- I Two display modes for preview: CIF, VGA.

### 6.2 Installing SceneDirector

#### 6.2.1 System Requirement

The current version of the tools are capable of running under Windows98® and Windows2000®. And it is recommended to run under Windows2000®. The propositional minimum system requirements are:

- CPU clock: PII or later



- Capacity of memory: 256 MB
- Free hard disk space: 500 MB

## 6.2.2 Installing

Taking the following steps, and you can install this tool on your computer:

1. Execute the installation file (.exe).
2. Follow on-screen prompts.

During installation, you will be asked to select one type of installation: **Typical**, **Compact** or **Custom**. SUNPLUS highly recommends you to install the tool with the "Typical" type which should apply for most of users. Selecting "Compact" results in the tool of minimal configuration installed on your computer. The "Custom" option is usually for advanced users.

Sunplus Confidential for SPECTRUM Use Only

---

---

## 7 SYSTEM OVERVIEW

---

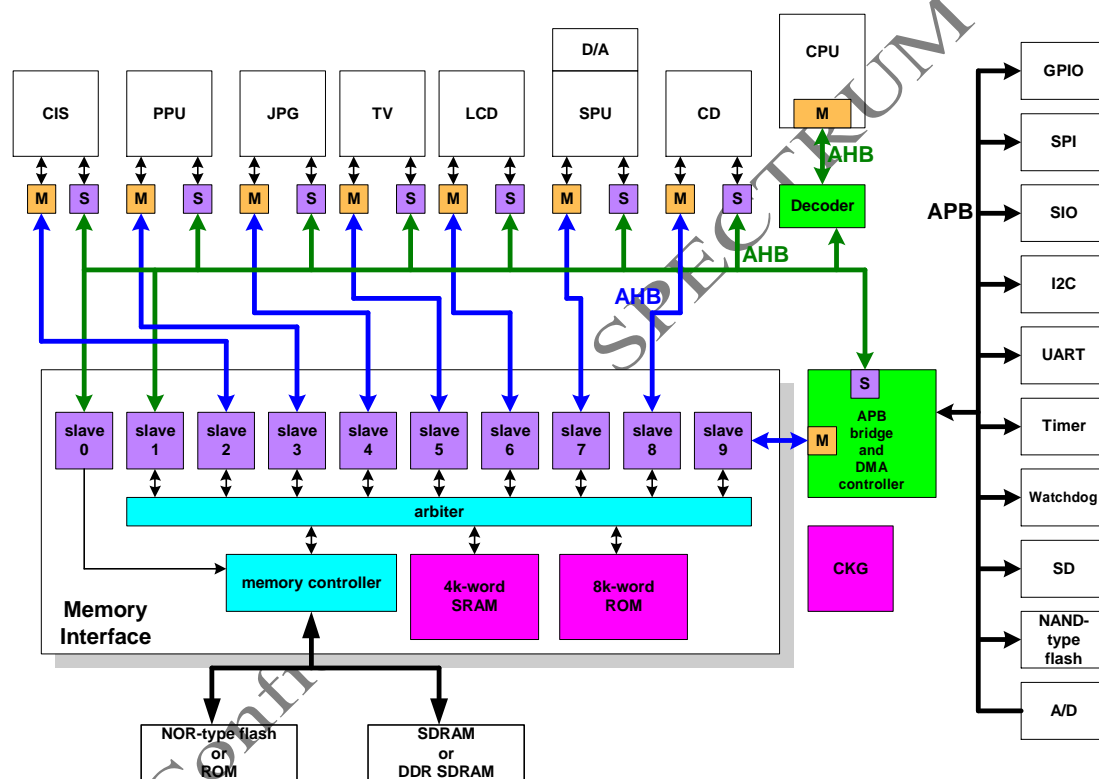
---

### 7.1 General Description

SPG290A, a highly integrated SOC consists of several processors, including high performance Sunplus 32bits RISC processor, **S+Core**, **2D Picture Processing Unit (PPU)**, 24 channel **Sound Processing Unit (SPU)**, **simple profile MP4 codec**, **multi-frames alpha-blanding DMA**, and also **embedded CD servo & LCD controller** to meet more richer multimedia entertainment plate form and variable portable products with one single chip solution, such as high quality interactive game consoles, portable learning aids or portable media players.

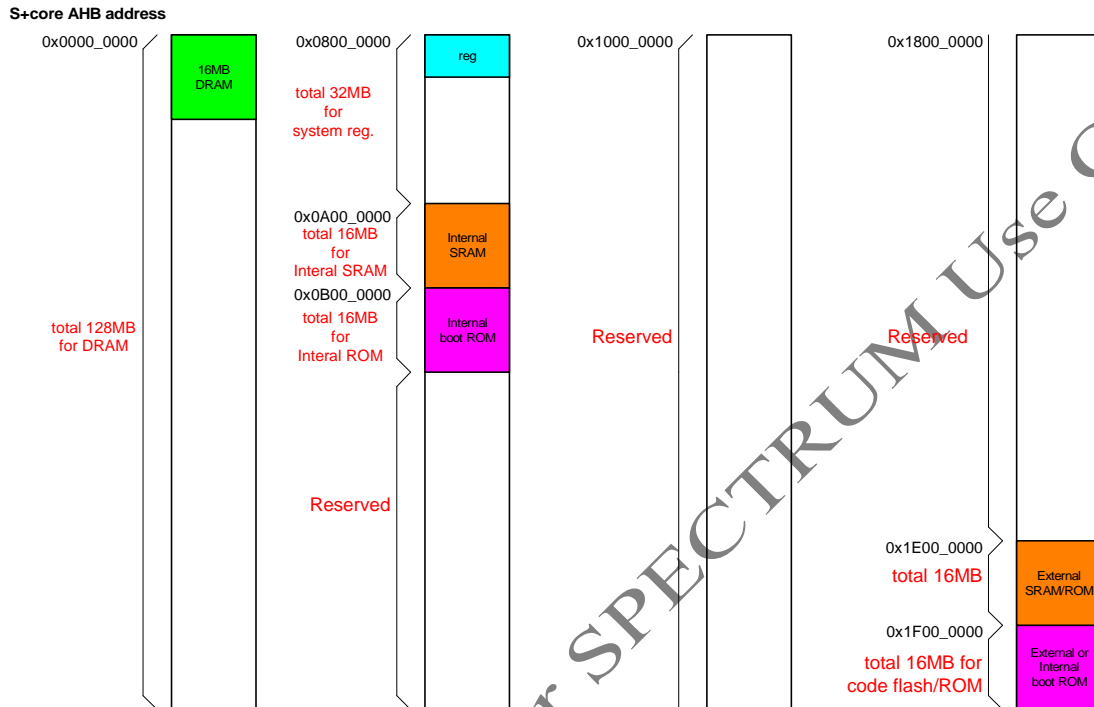
## 8 MEMORY INTERFACE UNIT - MIU

MIU supports several different types of external memories, SDRAM, EDO DRAM, Parallel ROM, and also Nand Flash to let users can use it to manage and design their application systems with more flexible selection of memory chips. In addition, MIU also manage 2 internal embedded memory blocks inside SPG290, 128K bits internal SRAM as **Local Data Memory** and 256K bits ROM as embedded **BOOT ROM**.



## 8.1 Memory Mapping

### MIU Memory Map for SPG290



## 8.2 System registers : Each module used 16k bytes range.

Range	Describe
0x0800_0000 ~ 0x0800_FFFF	CSI
0x0801_0000 ~ 0x0801_FFFF	PPU
0x0802_0000 ~ 0x0802_FFFF	JPG
0x0803_0000 ~ 0x0803_FFFF	TV
0x0804_0000 ~ 0x0804_FFFF	LCD
0x0805_0000 ~ 0x0805_FFFF	SPU
0x0806_0000 ~ 0x0806_FFFF	CD
0x0807_0000 ~ 0x0807_FFFF	MIU(system reg)
0x0808_0000 ~ 0x0808_FFFF	APBDMA(system reg)
0x0809_0000 ~ 0x0809_FFFF	BUFCTL
0x080A_0000 ~ 0x080A_FFFF	IRQCTL
0x080B_0000 ~ 0x080B_FFFF	GPUBUF
0x080C_0000 ~ 0x080C_FFFF	LDMDMA
0x080D_0000 ~ 0x080D_FFFF	BLNDMA
0x080E_0000 ~ 0x080E_FFFF	TPGBUF
0x080F_0000 ~ 0x080F_FFFF	AHBDEC
0x0810_0000 ~ 0x0810_FFFF	GPIO
0x0811_0000 ~ 0x0811_FFFF	SPI
0x0812_0000 ~ 0x0812_FFFF	SIO
0x0813_0000 ~ 0x0813_FFFF	I2C

0x0814_0000 ~ 0x0814_FFFF	I2S
0x0815_0000 ~ 0x0815_FFFF	UART
0x0816_0000 ~ 0x0816_0FFF	TIMER1
0x0816_1000 ~ 0x0816_1FFF	TIMER2
0x0816_2000 ~ 0x0816_2FFF	TIMER3
0x0816_3000 ~ 0x0816_3FFF	TIMER4
0x0816_4000 ~ 0x0816_4FFF	TIMER5
0x0816_5000 ~ 0x0816_5FFF	TIMER6
0x0816_6000 ~ 0x0816_6FFF	RTC
0x0817_0000 ~ 0x0817_FFFF	WDOG
0x0818_0000 ~ 0x0818_FFFF	SD
0x0819_0000 ~ 0x0819_FFFF	FLASH
0x081A_0000 ~ 0x081A_FFFF	ADC
0x081B_0000 ~ 0x081B_FFFF	USB device
0x081C_0000 ~ 0x081C_FFFF	USB host
0x081D_0000 ~ 0x081D_FFFF	reserved
0x081E_0000 ~ 0x081E_FFFF	Reserved
0x081F_0000 ~ 0x081F_FFFF	reserved
0x0820_0000 ~ 0x0820_FFFF	SFTCFG
0x0821_0000 ~ 0x0821_FFFF	CKG
0x0822_0000 ~ 0x0822_FFFF	MP4
0x0823_0000 ~ 0x0823_FFFF	MIU2(system reg)
0x0824_0000 ~ 0x0824_FFFF	ECC

### 8.3 Control Registers

MIU have some control registers, that's registers is read/write by AHB slave 0 by CPU.

#### 8.3.1 CSI frame buffer start address

NAME	ADDR	D23-D8
P_TV_START_ADR1	0x88070044	TV_START_ADR1
P_TV_START_ADR2	0x88070048	TV_START_ADR2
P_TV_START_ADR3	0x88070064	TV_START_ADR3

TV\_START\_ADRx: The start address of TV frame buffer x

#### 8.3.2 PPU frame buffer start address

NAME	ADDR	D23-D8
P_LCD_START_ADR1	0x8807004C	PPU_LCD_ADR1
P_LCD_START_ADR2	0x88070050	PPU_LCD_ADR2
P_LCD_START_ADR3	0x88070068	PPU_LCD_ADR3

LCD\_START\_ADRx: The start address of LCD frame buffer x

#### 8.3.3 P\_SPU\_START\_ADR(0x88070054)

NAME	D23-D8
P_SPU_START_ADR1	PPU_SPU_ADR1

PPU\_SPU\_ADR1: The start address of SPU buffer



## 8.3.4 P\_SDRAM\_POWER\_DOWN (0x8807005C)

NAME	D3	D2	D1	D0
P_SDRAM_POWER_DOWN	DEEP	REDO_MRS	POWER_DW	SELF_REF

SELF\_REF: To force SDRAM to enter self-refresh

0: Disable

1: Enable

POWER\_DW: To force SDRAM to enter power-down

0: Disable

1: Enable

REDO\_MRS: To Re-Do MRS

0: Disable

1: Enable

DEEP: To force SDRAM to enter deep power-down

0: Disable

1: Enable

## 8.3.5 P\_MIU1\_SDRAM\_SETTING(0x88070060)

NAME	D14-D11	D10-D3	D2	D1	D0
P_MIU1_SDRAM_SETTING	RFR_CYCL E_NUM	ATRFR_PE RIOD	CAS_LA_CY CLE	tRCD_CYCL E	tRP_CYCLE
NAME	D31	D30-D27	D26-D25	D24-D23	D22-D15
P_MIU1_SDRAM_SETTING	MIU_EN	SDCLK_SEL	SDRAM_RN	SDRAM_CW	SFRFR_PE RIOD

tRP\_CYCLE: tRP cycles setup

0: tRP is 1 cycle

1: tRP is 2 cycles

tRCD\_CYCLE: tRCD cycles setup

0: tRCD is 1 cycle

1: tRCD is 2 cycles

CAS\_LA\_CYCLE: Cas latency cycles setup

0: Cas Latency is 2 cycles

1: Cas Latency is 3 cycles

ATRFR\_PERIOD: Every( ATRFR\_PERIOD x 16) cycles, a auto-refresh is issued

RFR\_CYCLE\_NUM: Decide how many cycles a auto-refresh is issued

SFRFR\_PERIOD: If ADG\_REQ doesn't occur for (SFRFR\_PERIOD x 256 x ATRFR\_PERIOD x 16) cycles, a hardware-triggered self-refresh is issued

SDRAM\_CW: SDRAM Col Width

00: 8bits word SDRAM is used

01: 16bits word SDRAM is used

SDRAM\_RN: 10: 32bits word SDRAM is used  
SDRAM Row Number  
00: 1024 rows per bank  
01: 2048 rows per bank  
10: 4096 rows per bank  
11: 8192 rows per bank

SDCLK\_SEL: SDRAM clock selection  
Select the phase delay of SDRAM\_CLK

MIU\_EN: MIU enable setup  
0: Disable  
1: Enable

### 8.3.6 P\_MIU\_STATUS(0x8807006C)

NAME	D0
P_MIU1_SDRAM_SETTING	MIU_STATUS
MIU_STATUS:	0: Non-occurred 1: SDRAM is in the self-refresh mode or power-down mode

### 8.3.7 MP4 RAW Image frame buffer

NAME	ADDR	D23-D8	D7-D0
P_MP4RAW_START_ADR1	0x88070070	MP4RAW_START_ADR1	
P_MP4RAW_START_ADR2	0x88070074	MP4RAW_START_ADR2	
P_MP4RAW_START_ADR3	0x88070078	MP4RAW_START_ADR3	

MP4RAW\_START\_ADRx: Start address of MP4 raw image frame buffer x

### 8.3.8 MP4 write frame buffer

NAME	ADDR	D23-D8	D7-D0
P_MP4W_START_ADR1	0x8807007C	MP4W_START_ADR1	
P_MP4W_START_ADR2	0x88070080	MP4W_START_ADR2	
P_MP4W_START_ADR3	0x88070084	MP4W_START_ADR3	

MP4W\_START\_ADRx: Start address of MP4 write frame buffer x

### 8.3.9 MP4 VLC buffer

NAME	ADDR	D23-D10	D9-D0
P_MP4V_START_ADR1	0x88070088	MP4V_START_ADR1	
P_MP4V_START_ADR2	0x8807008C	MP4V_START_ADR2	

MP4V\_START\_ADRx: Start address of MP4 VLC buffer x

### 8.3.10 P\_MP4\_FRAME\_BUF\_HSIZE(0x88070090)

NAME	D9-D2	D1-D0
P_MP4_FRAME_BUF_HSIZE	MP4_FB_HSIZE	





MP4\_FB\_HSIZE: MP4 Frame Buffer H-Size  
The number of pixels on a scan line.

### 8.3.11 P\_SDRAM\_SETTING2(0x88070094)

NAME	D1	D0
P_SDRAM_SETTING2	tRCD_CYCLE	tRP_CYCLE

tRP\_CYCLE: tRP cycles setup  
0: tRP is 1 cycle  
1: tRP is 3 cycles  
tRCD\_CYCLE: tRCD cycles setup  
0: tRCD is 2 cycle  
1: tRCD is 3 cycles

### 8.4 Arbiter

Memory request priority is TV > LCD > CSI, other mode priority decide by round-robin.

---

## 9 2D PICTURE PROCESS UNIT - PPU

---

PPU, Picture Process Unit, is a character base engine to render 512 foreground animation SPRITE and 3 background layers which are named TEXT in SPG290 system into a specified rectangle area for displaying to TV screen in the main external DRAM area. The PPU engine of SPG290 is inherit from SPG220 and enhance its computing efficiency to let SPG290 PPU can render up to 30 frames per second with 512 SPRITEs and 3 16bits colors TEXT layers in VGA mode. In addition, a 64 grade Alpha-Blending computing engine is included in SPG290 PPU, which can support each TEXT with different factor settings to let Game developers can add more special visual effect on the screen without slowing down the frame rate. The major features are listed as following:

- \* Support 3 screen modes:  
VGA (640x480 interlaced),  
QVGA (320x240 interlaced and progressive ),  
HVGA screen mode (640x240 interlaced and progressive).
- \* 3 TEXT layers with 64 order TEXT to TEXT alpha blending operation, 4/16/64/256 index color modes with characters and 16 bits BITMAP modes. The rectangular TEXT window is more larger than the physical screen size, so it can do screen scrolling more easily by changing the left-top coordinate of screen mapping on the TEXT window. The TEXT window size is dependent with screen mode. When VGA mode, 3 TEXT windows will be 1024 pixels wide and 512 pixels high, 512 pixels wide and 256 pixels high in QVGA mode.
- \* Variable Character Size  
Size (H x V) 8x8 / 8x16 / 8x32 / 8x64  
16x8 / 16x16 / 16x32 / 16x64  
32x8 / 32x16 / 32x32 / 32x64  
64x8 / 64x16 / 64x32 / 64x64 pixels
- \* 512 SPRITE memory entries, each of them can have different SPRITE size setting, any one of above 16 variant character size table. The color mode of each SPRITE is also can be different from others. In addition, with frame buffer architecture, there is no limitation of SPRITE numbers can be shown in a frame or a line simultaneously.
- \* Total 512 palette entries for indexed color and each of them can be a transparent color index.
- \* TEXT Horizontal lines Movement Control

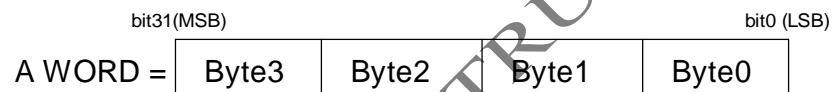
\* TEXT Compression / Extension function

## 9.1 Order of bytes

All data access in SPG290 32 bits system are **Little Endian**, including PPU. The most significant byte in a WORD, 32 bits data, is at the last place of a 4 bytes string.

Address or File order

Byte0
Byte1
Byte2
Byte3
Byte4



## 9.2 Color Mode

There are 6 color modes are available in SPG290. 4 modes with indexed color method, they are 2, 4, 6, 8 bits per pixel modes or 4, 16, 64, 256 indexed color mode. In BITMAP mode, 2 kinds of RGB pixel format can be applied, one is ARGB(1:5:5:5) where A at MSB is the enable bit of transparency and the other is RGB565(5:6:5).

### 9.2.1 Color modes of SPRITE

SPRITE consists of Characters and 4/16/64/256 indexed color mode, that is BITMAP mode is not available with SPRITE.

### 9.2.2 Color modes of TEXT

TEXT can support both Character mode and BITMAP mode. With Character mode, 4/16/64/256 indexed color modes are all available. A TEXT with BITMAP mode means the frame buffer of TEXT is no longer filled with Character data, instead each 16 bits in it represent a physical pixel's color data. There are 2 types of RGB pixel format defined in SPG290, one is ARGB and the other is RGB565.

### 9.3 Palette Memory

Pixel data of a character or a SPRITE are all Index numbers of Palette, not a real RGB color value. With different color modes, 4, 16, 64, or 256 colors, pixel formats in bits are also different. For example, in 16 color mode character, each pixel needs 4 bits, and 8 bits when 256 color mode. Palette Memory is a block of internal static RAM in SPG290 with 16 bits per Palette and totally 512 entries. In addition, with different color modes, Palette Memory can be segmented into several Banks so as let users can manage it more efficiently. The Palette Bank is also used in TEXT structure or attribute and SPRITE structure to calculate the real color address in Palette RAM.

#### 9.3.1 Palette Bank

Color modes	Total Palette Banks
2 bits 4 color mode	32
4 bits 16 color mode	32
6 bits 64 color mode	8
8 bits 256 color mode	2

Physical Palette address =  $0x81001000 + \text{Palette Offset}$  ;

Palette Offset =  $(\text{Palette Bank} \times \text{Color Mode}(4/16/64/256) + \text{Pixel Color Index}) \times 4$

\* SPG290 use 4 bits to store 4 color mode pixel like 16 color mode

**Palette Bank Registers in PPU :**

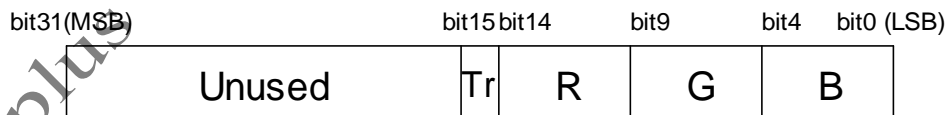
		B12	B11	B10	B9	B8
P_Tx1_attribute	0x8801_0028	Palette Bank of TEXT1				
P_Tx2_attribute	0x8801_0044	Palette Bank of TEXT2				
P_Tx3_attribute	0x8801_0060	Palette Bank of TEXT3				
P_Sprite(N)_Attribute	$0x8801\_4004 + (8 \times N)$	Palette Bank of Sprite(N)				

#### 9.3.2 Pixel format of Palette

Each Palette entry is a 32 bit word, but only last significant half word is valid for storing color values.

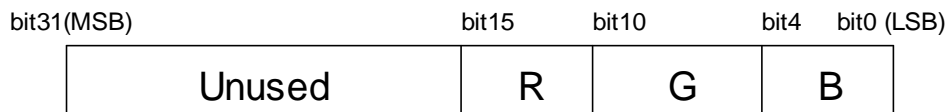
There are two RGB color pixel format defined in SPG290.

RGB555 mode:



Tr(bit15) : Transparent Bit. 1 : Transparent Color ; 0 : Solid Color

RGB565 mode:



	bit31(MSB)	bit15	bit14	bit9	bit4	bit0 (LSB)
Palette0 (0x8801 1000)	Unused	Tr	R	G	B	
Palette1 (0x8801 1000)	Unused	Tr	R	G	B	
Palette2 (0x8801 1000)	Unused	Tr	R	G	B	
Palette510 (0x8801 17F8)	Unused	Tr	R	G	B	
Palette511 (0x8801 17FC)	Unused	Tr	R	G	B	

## 9.4 Attributes

Each character/sprite comprises many attributes including color mode, depth, horizontal / vertical size, horizontal / vertical flip, blend and palette bank selection.

### 9.4.1 Palette

Denote the Palette Bank of Character, SPRITE or TEXT.

		B12	B11	B10	B9	B8
P_Tx1_attribute	0x8801_0028	Palette Bank of TEXT1				
P_Tx2_attribute	0x8801_0044	Palette Bank of TEXT2				
P_Tx3_attribute	0x8801_0060	Palette Bank of TEXT3				
P_Sprite(N)_Attribute	0x8801_4004+(8*N)	Palette Bank of Sprite(N)				

### 9.4.2 Depth

Depth defines the relevant position for characters and layers. There are four depth layers, four for the sprites and the rest for the text, as shown in the figure. The layer with larger depth number will be displayed when two elements are overlapped in different depth. If Text1 and Text2 and Text3 are located at the same layer, the Text3 would be dominated in the overlap area. If two or more sprites are located at the same layer, the sprite with larger sprite memory

address number is also dominated in the overlap area. In the figure below, the priority of sprite and text is: Sprite Layer3 > Text Layer3 > Sprite Layer 2 > Text Layer2 > Sprite Layer 1 > Text Layer 1 > Sprite Layer 0 > Text Layer 0 where ">" means on top.

		B14	B13
P_Tx1_attribute	0x8801_0028	Depth1	
P_Tx2_attribute	0x8801_0044	Depth2	
P_Tx3_attribute	0x8801_0060	Depth3	
P_Sprite(N)_Attribute	0x8801_4004+(8*N)	Sprite(N)_Depth	

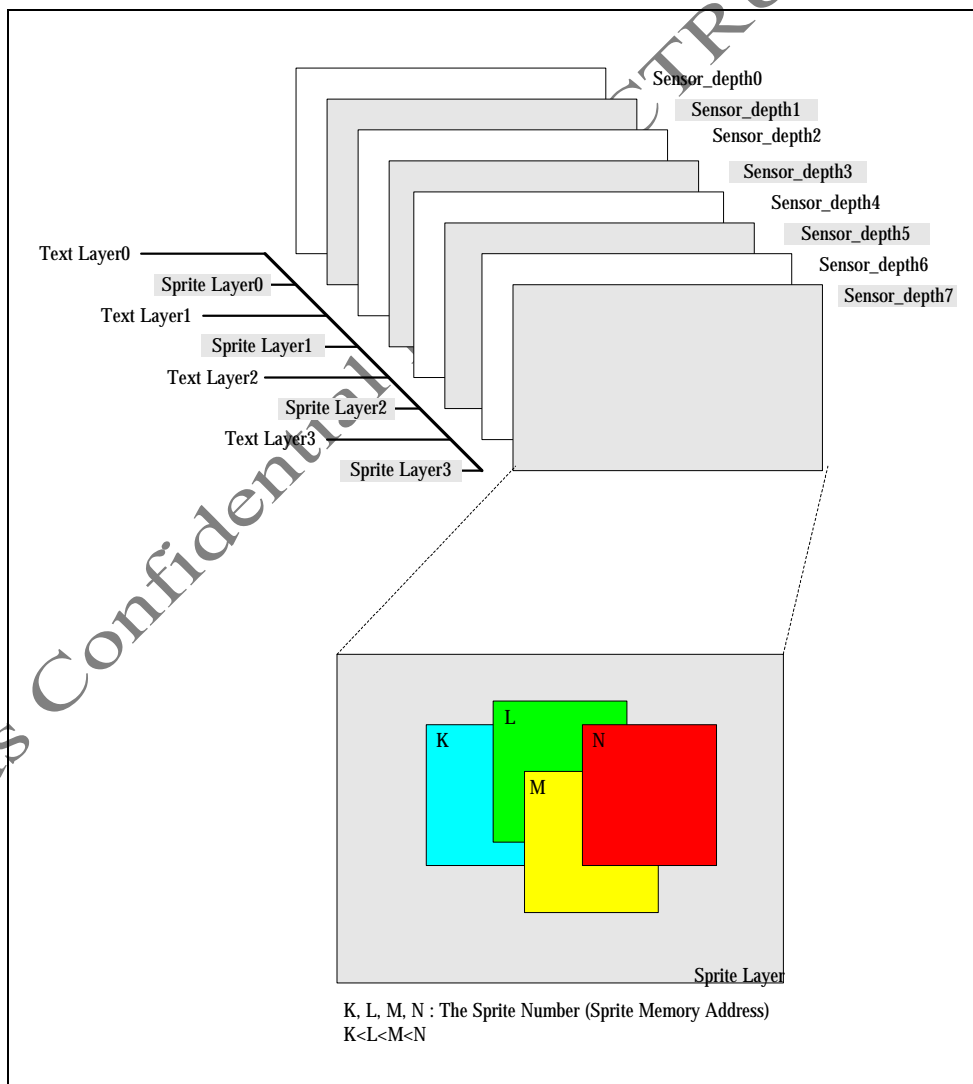
**Depth1:** Text1 Depth

**Depth2:** Text2 Depth

**Depth3:** Text3 Depth

**Sprite(N)\_Depth:** The Nth Sprite Depth

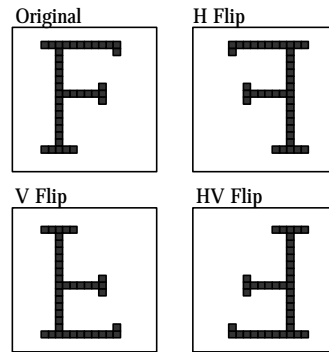
00:Deepest layer ~ 511    11:Shallowest layer



### 9.4.3 Character Flip

Flip function means the mirror effect. The vertical and horizontal flips can be controlled independently.

This function operates character by character individually. Please see the figures below for illustration.



		B3	B2	B1	B0
P_Tx1_attribute	0x8801_0028	Flip1			
P_Tx2_attribute	0x8801_0044	Flip2			
P_Tx3_attribute	0x8801_0060	Flip3			
P_Sprite(N)_Attribute	0x8801_4004+(8*N)	Sprite(N)_Flip			

**Flip1:** Text1 Character Flip

**Flip2:** Text2 Character Flip

**Flip3:** Text3 Character Flip

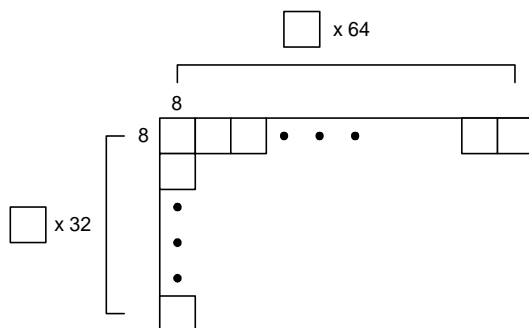
**Sprite(N)\_Flip:** Nth Sprite Flip

00: No Flip      01: Horizontal Flip

10: Vertical Flip    11: Vertical and Horizontal Flip

#### 9.4.4 Character Size

Character size defines the number of pixel for each character. Each sprite has its own character size, or in other words, a sprite can be different character sizes. In contrast, a text (Text1 ,Text2 or Text3) can only allow one character size in it. For example, suppose 8x8 character-size is chosen for a text screen. As a result, the entire text screen is all filled with 8x8 character-size.



The following character sizes are available in SPG290.

Vertical Size (pixel)	Horizontal Size (pixel)			
	8	16	32	64
8	8 x 8	8 x 16	8 x 32	8 x 64
16	16 x 8	16 x 16	16 x 32	16 x 64
32	32 x 8	32 x 16	32 x 32	32 x 64
64	64 x 8	64 x 16	64 x 32	64 x 64

		B7	B6	B5	B4
P_Tx1_attribute	0x8801_0028	Vs1		Hs1	
P_Tx2_attribute	0x8801_0044	Vs2		Hs2	
P_Tx3_attribute	0x8801_0060	Vs3		Hs3	
P_Sprite(N)_Attribute	0x8801_4004+(8*N)	Sprite(N)_vsize		Sprite(N)_hsize	

**Vs1:** Text1 Character Vertical Size

**Hs1:** Text1 Character Horizontal Size

**Vs2:** Text2 Character Vertical Size

**Hs2:** Text2 Character Horizontal Size

**Vs3:** Text3 Character Vertical Size

**Hs3:** Text3 Character Horizontal Size

**Sprite(N)\_vsize:** Nth Sprite Vertical Size

**Sprite(N)\_hsize:** Nth Sprite Horizontal Size

00: 8 pixels

01: 16 pixels

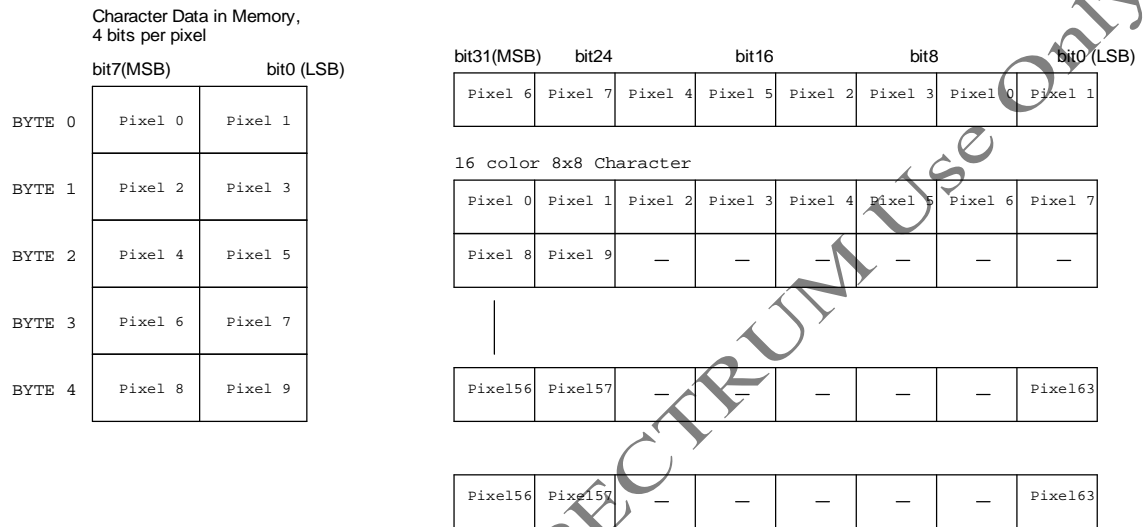
10: 32 pixels

11: 64 pixels



### 9.4.5 Character Data

Character data is a sequence of indexed color pixels in bytes. For example a 16 bits 8x8 character data stored in Frame Buffer memory are as following sample.

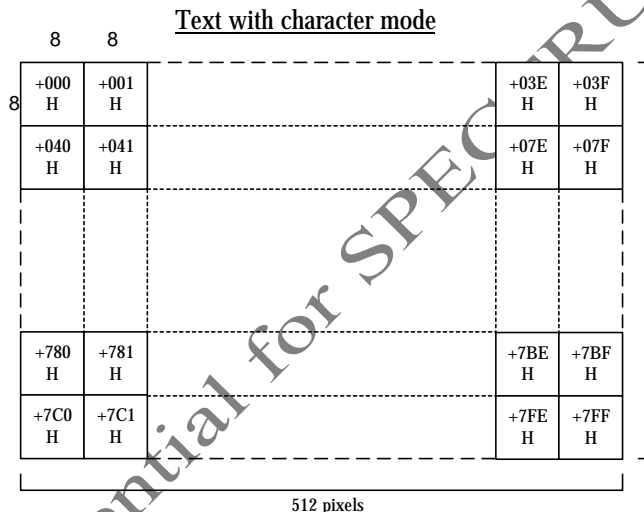


## 9.5 Text Screen

Text Screen consists of a set of characters (character mode) or lines (bitmap mode). The characters within a text must be the same size. This section describes the relationship between Text Memory content and Text Screen. The character mode and bitmap mode will be illustrated individually.

### 9.5.1 Character Mode

Text is formed by a set of characters. There are 16 character-size modes, but the size of a Text Screen is always 512 (H) x 256 (V) pixels in QVGA mode or 1024 (H) x 512 (V) pixels in VGA mode. The following diagram is an example with characters-size of 8x8 to form a text in QVGA mode.



### 9.5.2 Bitmap Mode

Text is formed by a set of pixels horizontally (from left to right), called bitmap mode. As a result, there are 512 horizontal pixels and 256 vertical pixels of cif mode and 1024 horizontal pixels and 512 vertical pixels of vga mode in bitmap mode. When this mode is used, the Vertical/Horizontal Flip Effect is not effective. Also, the bitmap addressing Mode is automatically chosen. In bitmap addressing Mode, the pointer and attribute arrays store the address of each line. Each line has its pointer and attribute. The three text layers can be chosen as the bitmap mode or the character mode independently.

		B3	B2	B1	B0
P_Tx1_control	0x8801_002c	Txe1			Linr1
P_Tx2_control	0x8801_0048	Txe2			Linr2
P_Tx3_control	0x8801_0064	Txe3			Linr3

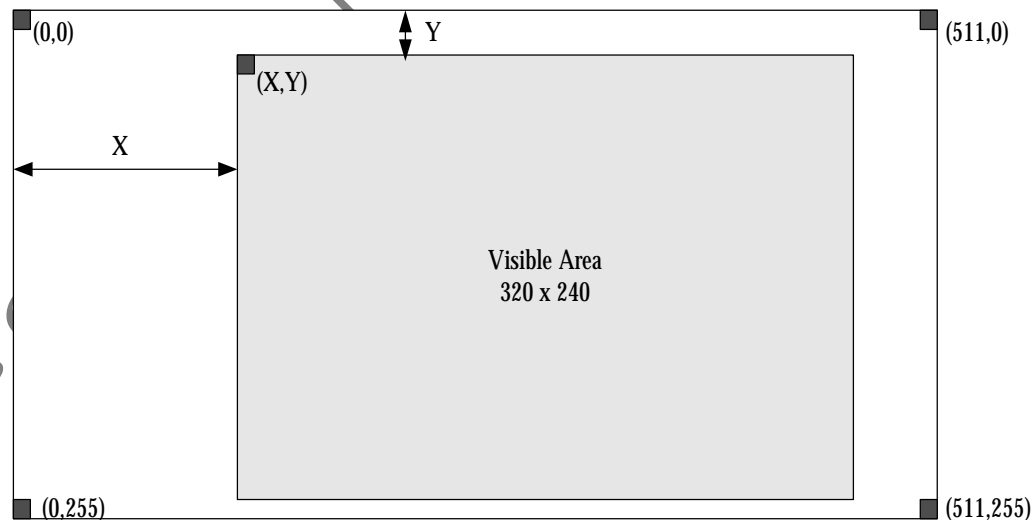
<b>Txe1:</b> Text1 Enable	0: Disable (Non Visible)	1: Enable (Visible)
<b>Linr1:</b> Text1 Bitmap Mode Enable	0: Character Mode	1: Bitmap Mode
<b>Txe2:</b> Text2 Enable	0: Disable (Non Visible)	1: Enable (Visible)
<b>Linr2:</b> Text2 Bitmap Mode Enable	0: Character Mode	1: Bitmap Mode
<b>Txe3:</b> Text3 Enable	0: Disable (Non Visible)	1: Enable (Visible)
<b>Linr3:</b> Text3 Bitmap Mode Enable	0: Character Mode	1: Bitmap Mode

## 9.6 Text Coordinate

The origin of Text Screen is located at the top-left corner. The area of Text Screen defined in the Text Memory is 512x256 in CIF Mode, while the visible area is 320x240. The (X, Y) defines the position of visible window, shown in the diagram.

		B8	B7	B6	B5	B4	B3	B2	B1	B0
P_Tx1_X_position	0x8801_0020	X1								
P_Tx1_Y_position	0x8801_0024	Y1								
P_Tx2_X_position	0x8801_003c	X2								
P_Tx2_Y_position	0x8801_0040	Y2								
P_Tx3_X_position	0x8801_0058	X3								
P_Tx3_Y_position	0x8801_005c	Y3								

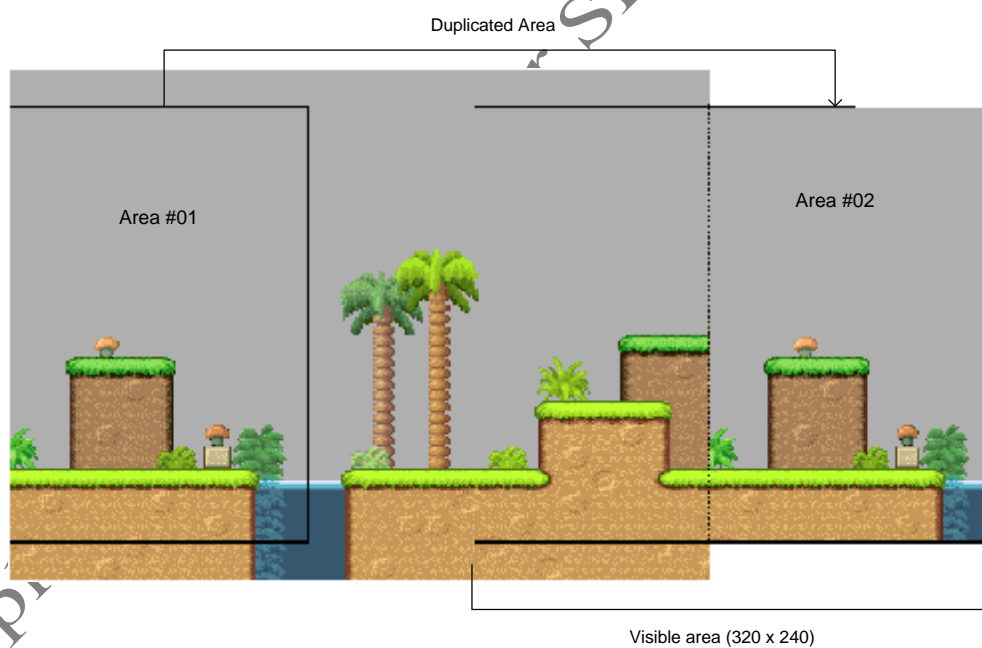
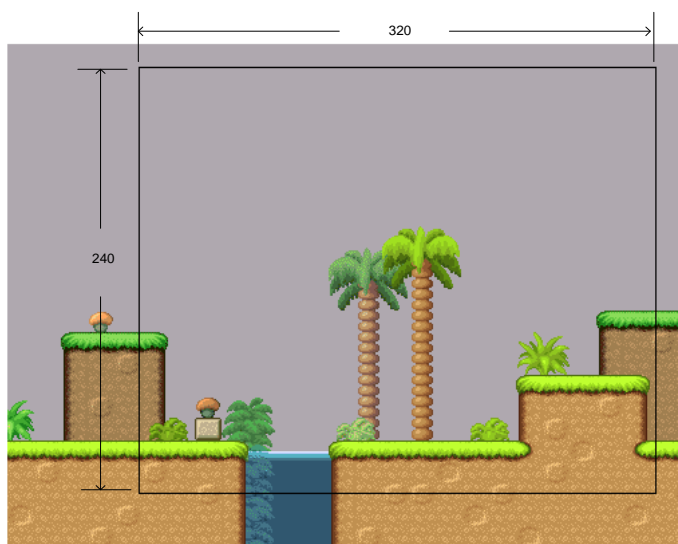
Text Coordinate (X,Y)

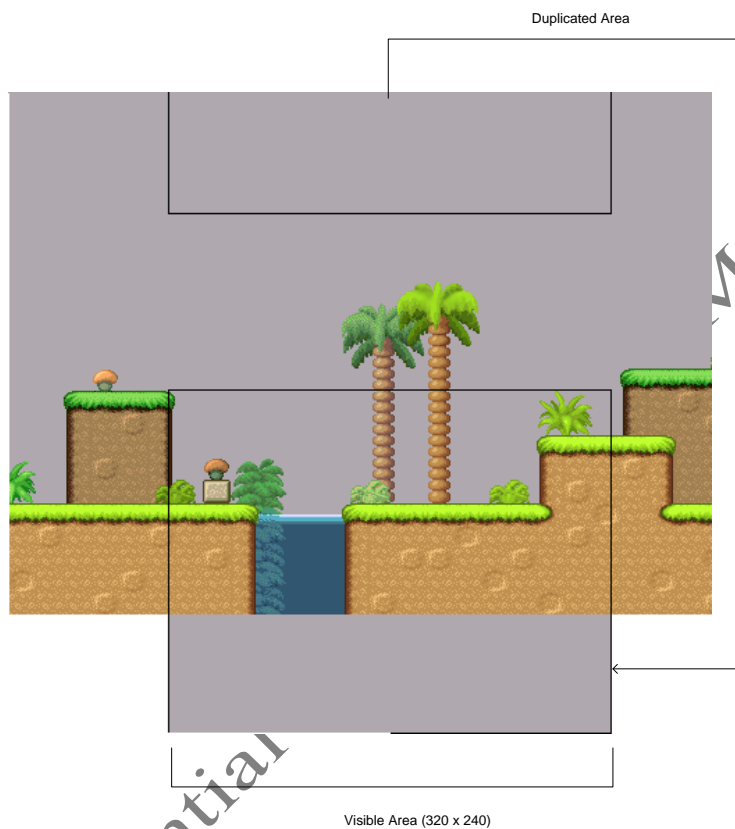


Text Screen loops up-down and left-right sides.

In Fig. 1, the visible area is 320 x 240 and the entire screen is 512 x 256 pixels. As the visible area

moves to right and exceeds to the screen view, the background on the left (Area #01) will be duplicated to the right (Area #02), see Fig.2. Similarly, when the visible area moves down and exceeds the screen view, the background on the top will be duplicated to the bottom, see Fig. 3.

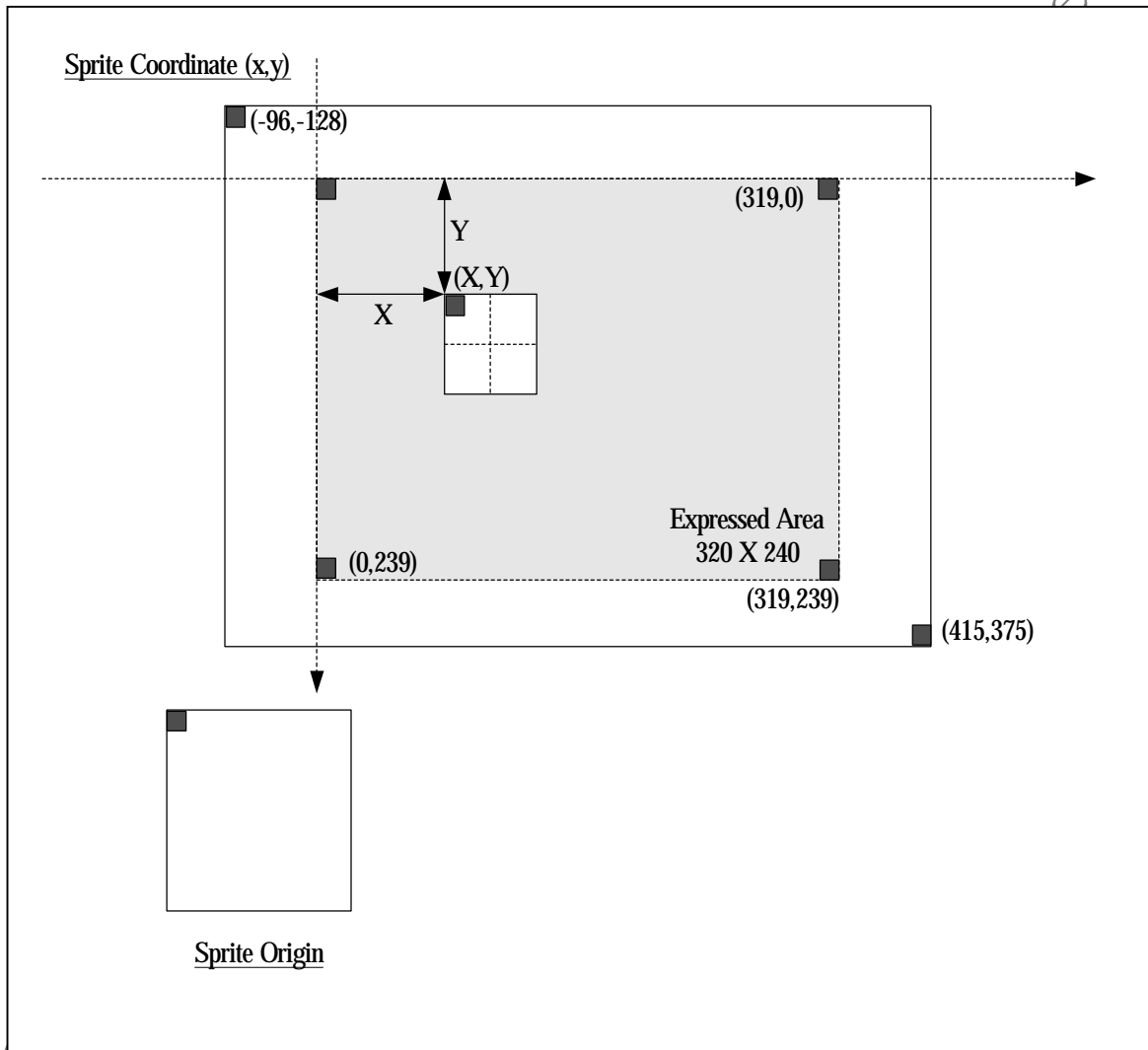




### 9.7 Sprite Coordinate

The origin of the sprite screen is at the left top corner of the visible area. The coordinate origin of sprite is at the left top of the sprite, as shown in the diagram.

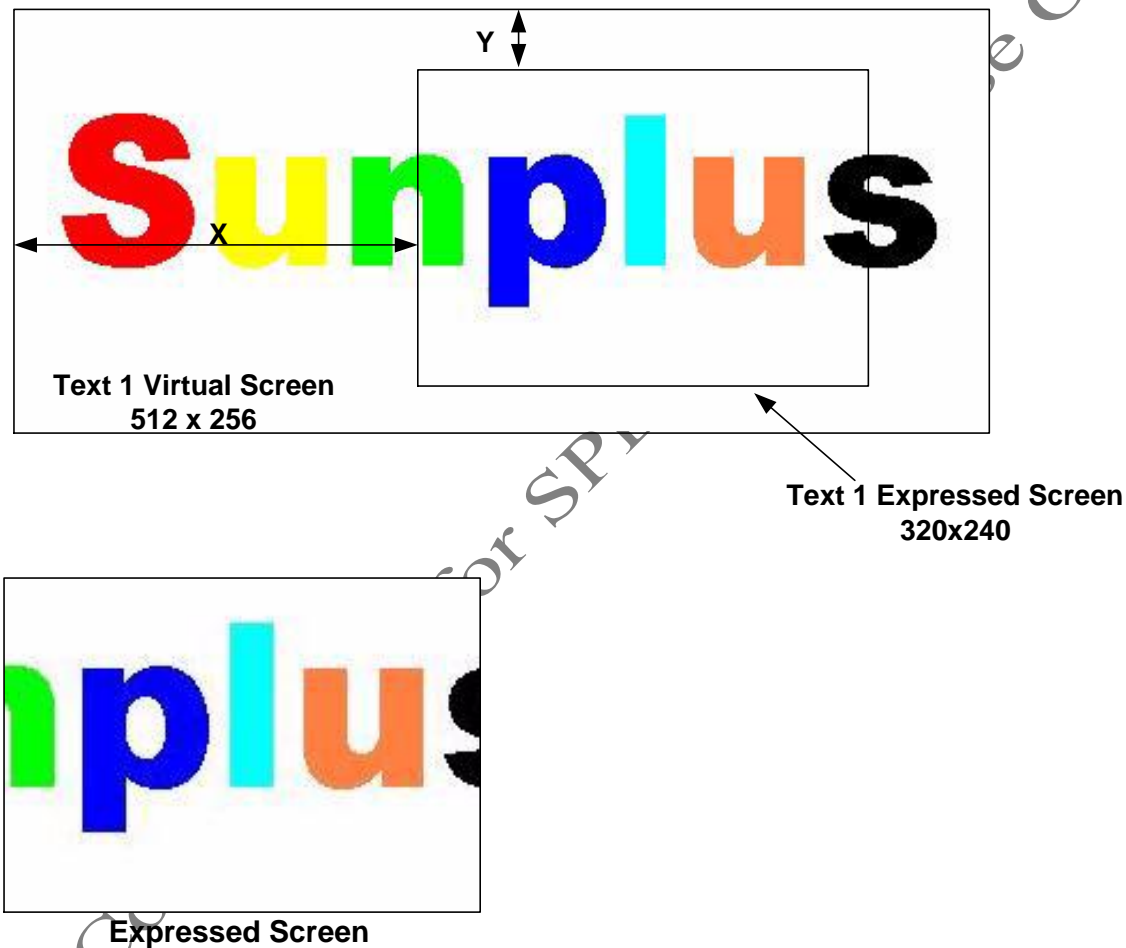
		D9	B8	B7	B6	B5	B4	B3	B2	B1	B0
P_Sprite(N)_X	$0x8801\_4002+(8*N)$	Sprite(N)_X									
P_Sprite(N)_Y	$0x8801\_4006+(8*N)$	Sprite(N)_Y									

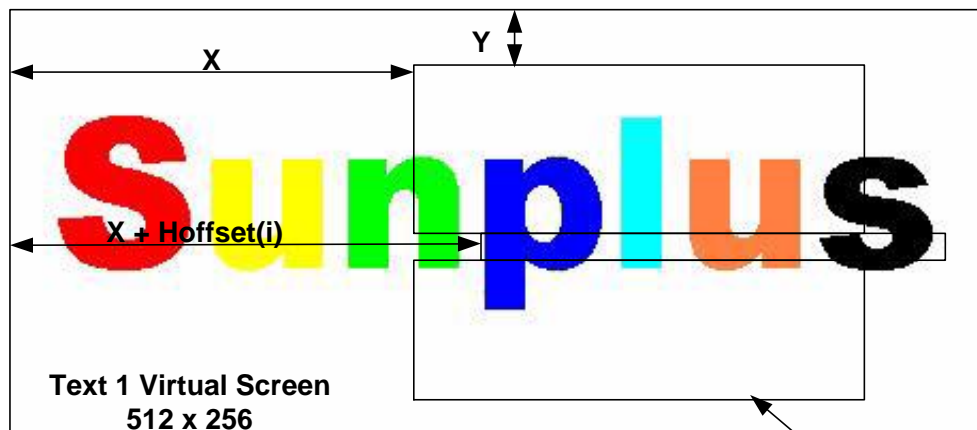


### 9.8 Text Horizontal Movement Control

The **Horizontal Movement Control** provides the line-based individual horizontal movement control.

Each horizontal scanning line has a horizontal movement offset value. The offset value will be added to the Text Position, described in the **Text Coordinate**.





**Text 1 Virtual Screen**  
512 x 256

**Text 1 Expressed Screen**  
320x240



**Expressed Screen**



These control registers Offset(i) are between 0x8801\_2000 to 0x8801\_27FC.

	B9 - b0
0x8801_2000	Offset0
0x8801_2004	Offset1
0x8801_27FC	Offset511

0x8801\_002c

b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	---	---	MVEN1	---	---	---	---

0x8801\_0048

b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	---	---	MVEN2	---	---	---	---

0x8801\_0064

b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	---	---	MVEN3	---	---	---	---

The MVEN is the switch for Horizontal movement control. When MVEN is given "high", the offset value would be added to the Horizontal Position.

## 9.9 Text Compression/Extension Control

The **Text Compression Control** provides vertical compression (VCMP) and line-based individual horizontal compression (HCMP) controls. The whole text has a vertical compression value while each horizontal scanning line has its horizontal compression value.

### 9.9.1 Vertical compression

For vertical compression, VCMP1 & VCMP2 & VCMP3 in control registers 0x8801\_002c and 0x8801\_0048 and 0x8801\_0064 are the switches to enable/disable vertical compression. Moreover, the compression scale is given in 0x8801\_0074, and the line compression step is given in 0x8801\_0078. This scale and step are available for text1, text2, text3 or both text1 and text2 and text3, depending on the setting of VCMP1 & VCMP2 & VCMP3. However, there is only one scale for whole text.

0x8801\_002c

b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	VCMP1	---	---	---	---	---	---

0x8801\_0048

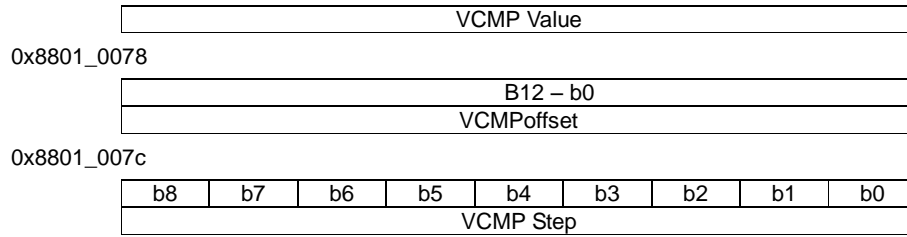
b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	VCMP2	---	---	---	---	---	---

0x8801\_0064

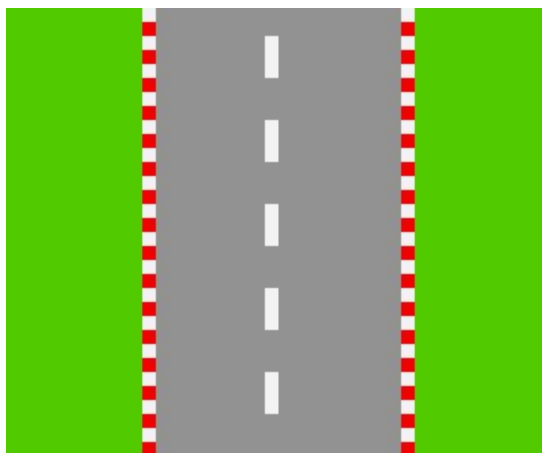
b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	VCMP3	---	---	---	---	---	---

0x8801\_0074

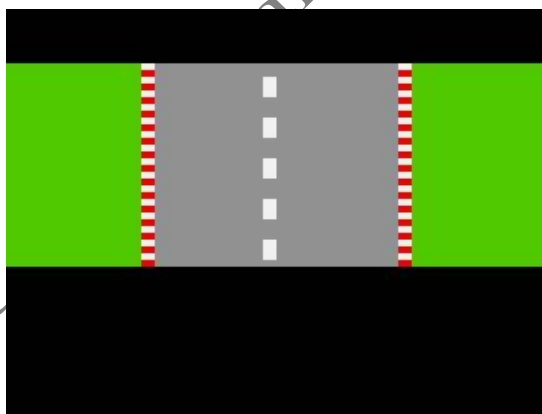
b8	b7	b6	b5	b4	b3	b2	b1	b0
----	----	----	----	----	----	----	----	----



Note: When VCMP is given “high”, the corresponding text would be compressed by the scale set in 0x8801\_0074. “20h” means no compression; “40h” represents double compression and “10h” expands the size to double.

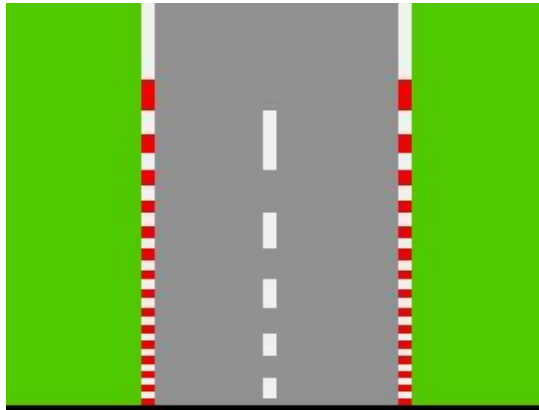


Vcmp\_Value = 20H  
Vcmp\_Step = 0H  
Vcmp\_Y\_offset = 0H

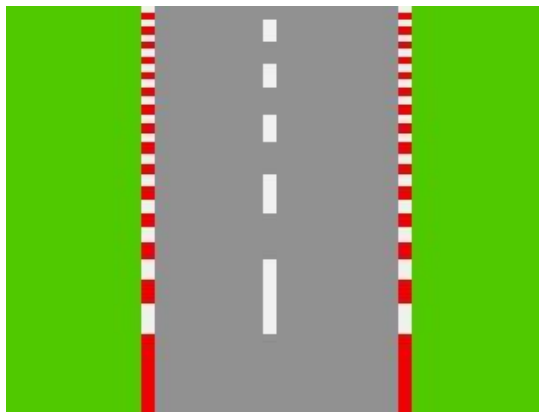


↓  
Vcmp\_Y\_offset = 20H  
↑

Vcmp\_Value = 40H  
Vcmp\_Step = 0H  
Vcmp\_Y\_offset = 20H



Vcmp\_Value = 0H  
Vcmp\_Step = 9H = +9  
Vcmp\_Y\_offset = 0H



Vcmp\_Value = 42H  
Vcmp\_Step = F7H = -9  
Vcmp\_Y\_offset = 0H

### 9.9.2 Horizontal compression

Different from vertical compression, users can have different compression scale for each Horizontal scanning line in horizontal compression. The HCMP in control register 0x8801\_002c is the switch to control horizontal compression. The scales are set in 0x8801\_3000~0x8801\_37FC. *Note that horizontal compression is only available in text1 and character mode.*

0x8801\_002c

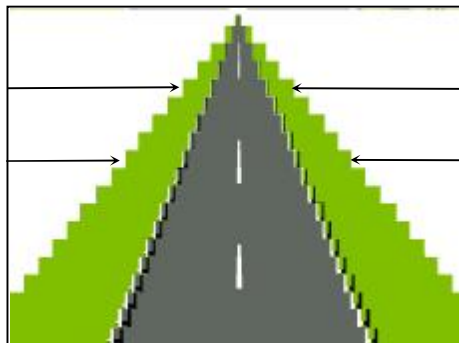
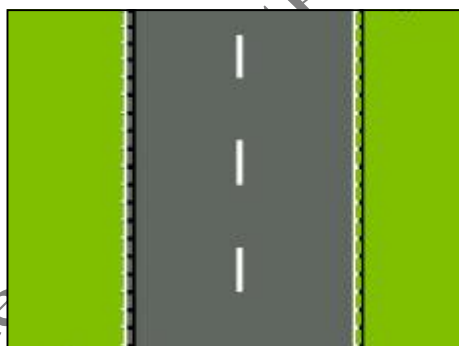
b8	b7	b6	b5	b4	b3	b2	b1	b0
---	---	---	HCMP	---	---	---	---	---

0x8801\_3000~0x8801\_37FC

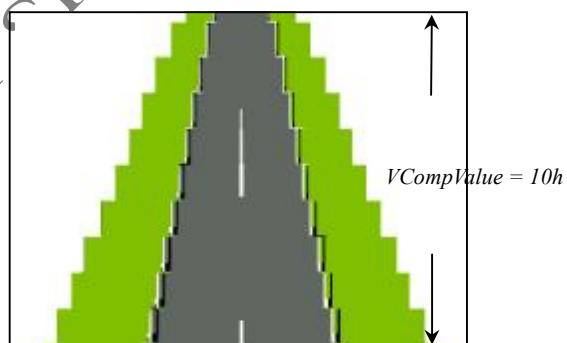
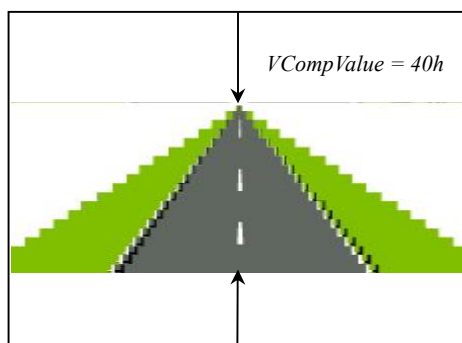
	B8 - b0
0x8801_3000	HcmpValue0
0x8801_3004	HcmpValue1
0x8801_37FC	HcmpValue511

Note: When HCMP is given "high", compression for text1 is active. Each horizontal scanning line is compressed according to the scale given in 0x8801\_3000~0x8801\_37FC. "80h" means no compression; "40h" represents double compression.

*Horizontal  
Compression*


*HcmpValue i = 20h*
*HcmpValue j = 40h*

Example:



### 9.9.3 Horizontal and Vertical Compression

Example:



### 9.9.4 Horizontal/Vertical Compression and Movement

Example:



### 9.10 Sprite Memory

Each sprite has its own parameters including **position**, **depth**, **palette**, **character size**, **number**, **flip**, **blending**, and **color mode**. There are 512 sprites stored in the Sprite Memory. CPU is able to access Sprite Memory directly. Every sprite contains four words to store parameters. The word1 stores the character number, word2 for storing the X position, word3 for storing the attributes, and word4 for storing Y position. The definition for each block is shown below.

Word1- Character Number

b15 – b0
Number

Word2 – X-Position

B25 – b16
PX

Word3 – Attribute

b15	b14	b13	b12	b11	b10	b9	b8
Blend	Depth		Palette Bank				
b7	b6	b5	b4	b3	b2	b1	b0
VSIZE		HSIZE		VFLIP	HFLIP	Color	

Word4 – Y-Position

B31-b26	B25 – b16
Blend_Level	PY

**Blend** Blending effect control

**Depth** Depth layer selection

**Palette** Color Palette Bank of the sprite.

**VSIZE / HSIZE** Vertical / horizontal character size.

**VFLIP / HFLIP** Vertical / horizontal flip.

**Color** Color Mode selection

**PX** Horizontal position range is between the (–512 ~ 511).

**PY** Vertical position range is between (–512 ~ 511).

**Number** Character number. Zero represents transparent character.

**Blend\_Level** Blend Level value range is between(0~63).

### 9.11 Text Memory

Text screen consists of characters or lines. In addition, the information stored in Text Memory is called **Text Data Array**, which includes attribute array and character number array. Attribute Array stores the attribute of each character/line such as blend, flip and palette. Character number array stores the character number which is used for the addressing mode to access the character/pixel information from Pattern Memory. Number Array is always necessary and Attribute Array is only required in the case of Bitmap Mode or Attribute Array mode (RegisterMode=0).



The number of CharNumber Array Block and Attribute Array Block are dependent on the Text Screen Character Size and the Text Screen Bitmap Mode.

CIF Mode:

Character /Bitmap	Character Vsize (Pixels)	Character Hsize (Pixels)	No. CharNumber Array (Words)	No. of Attribute Array (Words)
Character	8	8	2048	1024
Character	8	16	1024	512
Character	8	32	512	256
Character	8	64	256	128
Character	16	8	1024	512
Character	16	16	512	256
Character	16	32	256	128
Character	16	64	128	64
Character	32	8	512	256
Character	32	16	256	128
Character	32	32	128	64
Character	32	64	64	32
Character	64	8	256	128
Character	64	16	128	64
Character	64	32	64	32
Character	64	64	32	16
Bitmap	--	--	256	128

VGA Mode:

Character /Bitmap	Character Vsize (Pixels)	Character Hsize (Pixels)	No. CharNumber Array (Words)	No. of Attribute Array (Words)
Character	8	8	8192	1024x4
Character	8	16	4096	512 x4
Character	8	32	2048	256 x4
Character	8	64	1024	128 x4
Character	16	8	4096	512 x4
Character	16	16	2048	256 x4
Character	16	32	1024	128 x4
Character	16	64	512	64 x4
Character	32	8	2048	256 x4
Character	32	16	1024	128 x4
Character	32	32	512	64 x4
Character	32	64	256	32 x4
Character	64	8	1024	128 x4
Character	64	16	512	64 x4
Character	64	32	256	32 x4
Character	64	64	128	16 x4
Bitmap	--	--	1024	128 x4

Location of Text Array is controlled by **Text Array Pointer Registers**. This register appoints the Text Memory Address.



0x8801\_0030 Text1 Character Number Array Pointer

D31	D30	.....	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
NumberPointer1													

0x8801\_004c Text1 Character Number Array Pointer

D31	D30	.....	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
NumberPointer1													

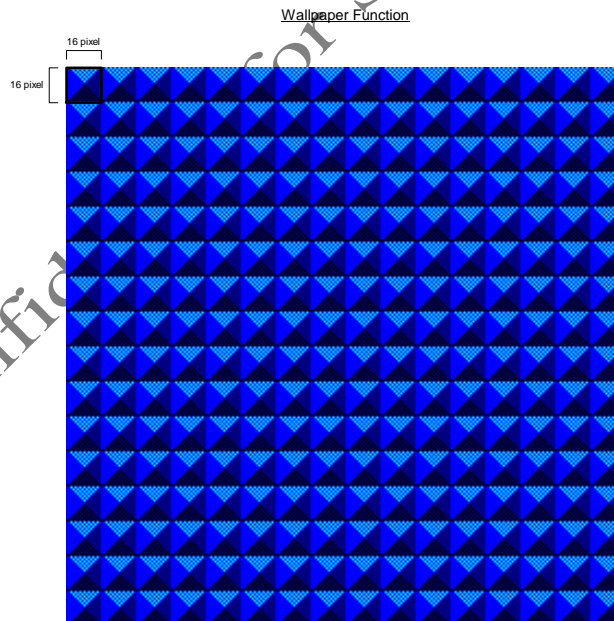
0x8801\_0068 Text1 Character Number Array Pointer

D31	D30	.....	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
NumberPointer1													

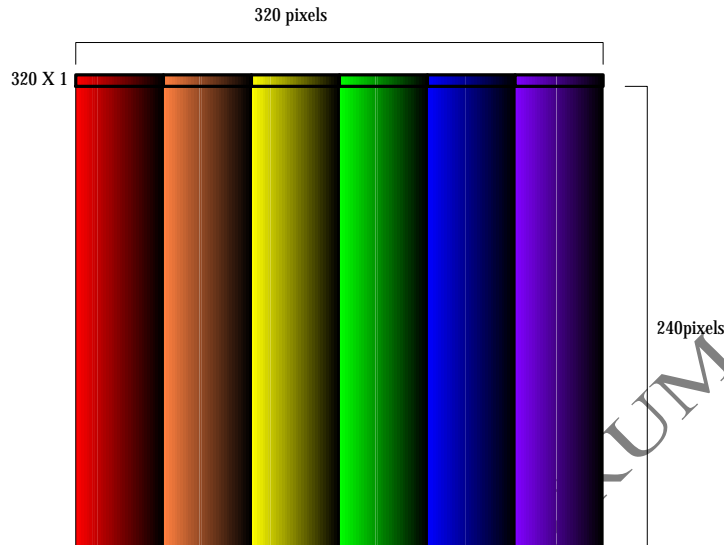
## 9.12 Wallpaper Mode

Wallpaper Mode loops single character or line (bitmap mode), as shown in the following figure. The attributes of the repeating character are the same as the 1<sup>st</sup> character/line defined in the Text Attribute.

### Character Mode



### Bitmap Mode



#### 9.13 Blending Effect

Blending effect is used to mix three pictures as shown in the following figures. The blending effect control of the sprite is the bit15 of the sprite attribute in the sprite memory. The text blending effect is controlled by the bit8 in 0x8801\_002c and 0x8801\_0048 and 0x8801\_0064 in register mode or bit6 in the attribute array in attribute mode. When the blending effect of a sprite or text is enabled, it will mix with the pixels that with the depth lower than it.

64 blending level can be selected by the Blend\_level at port 0x8801\_0038 and 0x8801\_0054 and 0x8801\_0070. and can be selected by the Blend\_level at each sprite attribute register(0x8804\_0006 + N\*8)

#### 9.14 32768/65536 High Color Mode

In SPG290 PPU, the 32768/65536 high color mode is provided to perform better image quality. This mode is controlled by the bit7/bit12 of 0x8801\_002c for text1 and 0x8801\_0048 for text2 and 0x8801\_0064 for text3. This mode only provides the bitmap mode. The programming methodology is similar to the bitmap mode.



## 9.15 Character Attributes

The following attribute parameters belong to Character for the Text Layer Character Mode and Sprites.

**Color Mode:** 4 / 16 / 64 / 256 colors per character

**Size:** 8x8 / 8x16 / 8x32 / 8x64  
16x8 / 16x16 / 16x32 / 16x64  
32x8 / 32x16 / 32x32 / 32x64  
64x8 / 64x16 / 64x32 / 64x64 pixel size

**Flip:** Vertical / Horizontal independent flip

**Palette:** Palette Bank

**Depth:** Depth control

**Blend:** Blending control

In case of Text screen, these parameters are stored in Text memory and Register set. In case of Sprite, they are stored in Sprite memory. Some parameters are defined in plural elements, and only one parameter is effective. This rule is described as follows.

### Text Screen

Color Mode:

Text Screen Color Mode in Register Set

Size :

Text Screen Character Size in Register Set

Flip :

if AddressMode = 0x

Flip Function Off

else if RegisterMode = 1

Text Screen Flip in Register Set

else

Attribute in Text Memory

Palette :

if RegisterMode = 1

Text Screen Palette Bank in Register Set

else

Attribute in Text Memory

Depth:

Text Screen Depth in Register Set

Blend :

if RegisterMode = 1

Text Screen Blending control in Register Set

else

Attribute in Text Memory



## Sprite

Color Mode:

Sprite Color Mode in Sprite Memory

Size:

Sprite Size in Sprite Memory

Flip:

Sprite Flip Control in Sprite Memory

Palette:

Sprite Palette Bank in Sprite Memory

Depth:

Sprite Depth in Sprite Memory

Blend:

Sprite Blending control in Sprite Memory

## 9.16 Bitmap Mode Attributes

The following attribute parameters belong to the Text Layer Bitmap Mode.

**Color Mode:** 4 / 16 / 64 / 256 colors per character

**Palette:** Palette Bank

**Depth:** Depth control

**Blend:** Blending control

**RGB555 Color Mode:** 32768 colors per pixel

**RGB565 Color Mode:** 65536 colors per pixel

In case of Text Screen, these parameters are stored in Register Set.

## Text Screen

Color Mode: Text Screen Color Mode in Register Set

Palette: Text Screen Palette Bank in Register Set

Depth: Text Screen Depth in Register Set

### 9.17 Addressing Mode

The 32-bit address pointer is required to access Pattern Data Block. The 32-bit address is sufficient to define any location in the memory map of picture processor or the CPU Memory. In any mode, the last 32-bit Address Pointer is generated through address conversion. The most suitable Addressing Mode must be chosen according to the specification.

- 1) Character Number Mode
- 2) Bitmap Address Mode

The Character Number Mode is for handling set of Characters and the Bitmap Address Mode is for the Bitmap Data.

**Picture Segment Registers** are necessary in address conversion. One Picture Segment Register consists of 32 bits, and 3 sets of Segment Register.



## 9.18 PPU API Services

ClearAllSetting( )

ClearAllSprite( )

MIUInit( )

InitColorPalette( )

InitSPColorPalette( )

BitmapShow( )

CharacterShow( )

WaitBlanking( )

PPU\_Service( )

Paint\_Sprite( )

EffectEnable( )

VCompressEffect( )

HCompressEffect(( )

HorizontalMovement( )

BlendEffect( )

FadeIn( )

FadeOut( )

Sunplus Confidential for SPECTRUM Use Only



---

**ClearAllSetting( )**

---

Prototype:                void ClearAllSetting(void)

Descriptions:            Initial all PPU register dependence variable value

Arguments:              (None)

Returns:                 (None)

---

**ClearAllSprite( )**

---

Prototype:               void ClearAllSprite(void)

Descriptions:            Clear all Sprite Buffer register dependence variable value

Arguments:              (None)

Returns:                 (None)

---

**MIUInit( )**

---

Prototype:               void MIUInit(UINT32 FB1, UINT32 FB2, UINT32 FB3)

Descriptions:            Initial MIU1 & MIU2 TV encode Frame buffer address and reset SDRAM

Arguments:

    FB1:                TV encode frame buffer 1 start address

    FB2:                TV encode frame buffer 2 start address

    FB3:                TV encode frame buffer 3 start address

Returns:                 (None)



## BitmapShow ( )

Prototype: void BitmapShow(INT32 nTextLayer, INT32 nXSize, INT32 nYSize, INT32 nColorMode, INT32 nBank, INT32 nDepthLayer, UINT32 \*PatAddr, UINT32 \*IndexAddr)

Descriptions: Show Texture for Bitmap mode

### Arguments:

nTextLayer: Select Text layer for Text1/Text2/Text3

#define TEXT1	0x00000000
#define TEXT2	0x00000001
#define TEXT3	0x00000002

nXSize: Texture Size for horizontal value

nYSize: Texture Size for vertical value

nColorMode: Text Color Mode

#define Color4	0x00000002
#define Color16	0x00000004
#define Color64	0x00000006
#define Color256	0x00000008
#define Color32768	0x00000010
#define Color65536	0x00000020

nBank: Palette Bank for 0~31

nDepthLayer: Text Depth Layer

#define TXDepth0	0x00000000
#define TXDepth1	0x00002000
#define TXDepth2	0x00004000
#define TXDepth3	0x00006000

\*PatAddr: The Texture data address

\*IndexAddr: The Texture index data address

Returns: (None)





## CharacterShow( )

Prototype: void CharacterShow(INT32 nTextLayer, INT32 nXSize, INT32 nYSize, UINT32 \*PatAddr, UINT32 \*IndexAddr, INT32 nColorMode, INT32 nCellXSize, INT32 nCellYSize, INT32 nPaletteBank, INT32 nDepth)

Descriptions: Show Texture for Character Mode

Arguments:

nTextLayer: Select Text layer for Text1/Text2/Text3

#define TEXT1	0x00000000
#define TEXT2	0x00000001
#define TEXT3	0x00000002

nXSize: Texture Size for horizontal value

nYSize: Texture Size for vertical value

\*PatAdr: The Texture data address

\*IndexAddr: The Texture index data address

nColorMode: Text Color Mode

#define Color4	0x00000002
#define Color16	0x00000004
#define Color64	0x00000006
#define Color256	0x00000008

nCellXSize: Character Size for Horizontal value

#define TXHSize8	0x00000000
#define TXHSize16	0x00000010
#define TXHSize32	0x00000020
#define TXHSize64	0x00000030

nCellYSize: Character Size for Vertical value

#define TXVSize8	0x00000000
#define TXVSize16	0x00000040
#define TXVSize32	0x00000080
#define TXVSize64	0x000000C0

nPaletteBank: Palette Bank for 0~31

nDepth: Text Depth Layer

#define TXDepth0	0x00000000
#define TXDepth1	0x00002000
#define TXDepth2	0x00004000
#define TXDepth3	0x00006000

Returns: (None)



## **InitColorPalette( )**

---

Prototype:                void InitColorPalette(UINT32 \*nPalAdr)

Descriptions:            Initial Background Color Palette

Arguments:

    \*nPalAdr:    The Color Palette data address

Returns:                (None)

## **InitSPColorPalette( )**

---

Prototype:                void InitSPColorPalette(UINT32 \*nPalAdr)

Descriptions:            Initial Sprite Color Palette

Arguments:

    \*nPalAdr:    The Color Palette data address

Returns:                (None)

## **WaitBlanking( )**

---

Prototype:                void WaitBlanking(void)

Descriptions:            Wait Vertical blanking start

Arguments:                (None)

Returns:                (None)

## **PPU\_Service( )**

---

Prototype:                void PPU\_Service(void)

Descriptions:            PPU Service function that called by IRQ53 to update Sprite, palette, Text move

Arguments:                (None)

Returns:                (None)



## Paint\_Sprite ( )

Prototype: void Paint\_Sprite (void)

Descriptions: Write Sprite Register to Sprite buffer, when IRQ53 run the PPU\_Service to update the Sprite Ram buffer, that will show Sprite to the screen

Arguments: (None)

Returns: (None)

## EffectEnable( )

Prototype: void EffectEnable(INT32 nText,INT32 nMode,INT32 nEnable)

Descriptions: Set Effect enable, like Blending/H-Compression/V-Compression/H-Movement

Arguments:

nText:	#define	TEXT1	0
	#define	TEXT2	1
	#define	TEXT3	2
nMode:	#define	TXMOVEN	0x00000010
	#define	TXHCMP	0x00000020
	#define	TXVCMP	0x00000040
	#define	BLEND	0x00000100
nEnable:	#define	ENABLE	0
	#define	DISABLE	1

Returns: (None)

---

**VCompressEffect( )**

---

Prototype:                void VCompressEffect(INT32 nCompValue,INT32 nCompStep,INT32 nCompOffset)

Descriptions:            Set vertical compression value when each TEXT have enable the V-CompressEffect

Arguments:

nCompValue:        vertical compression value  
nCompStep:         vertical compression step value  
nCompOffset:       vertical compression offset value

Returns:                (None)

---

**HCompressEffect( )**

---

Prototype:               void HCompressEffect(INT16 nMode,INT32 nCompValue,const INT32 \*ptrCmpTable)

Descriptions:            Set Horizontal compression value only for TEXT1

Arguments:

nMode                    Compression Mode  
0:    LINES\_SAME  
      NOTE:    If compress every line the same value,just write value to  
                 nCompValue.  
1:    LINES\_DIFFERENT  
      NOTE:    If compress every line the different value,you need to set up a  
                 value table.  
nCompValue:            Horizontal compression value  
\*ptrCmpTable:          Horizontal compression value table

Returns:                (None)

**HorizontalMovement ( )**

---

Prototype:                    void HorizontalMovement(INT32 StartScanline,INT32 EndScanline,INT32 \*IpScanline)

Descriptions:                Text layer every scanline change Horizontal Movement value by table

Arguments:

StartScanline:    from which Scanline to begin  
EndScanline:      from which Scanline to end  
\*IpScanline:       Scanline Movement data table

Returns:                    (None)

**BlendEffect( )**

---

Prototype:                    void BlendEffect(INT32 nText, INT32 nBlendValue)

Descriptions:                To set each TEXT the Blending value when Blending effect is enable

Arguments:

nText:                    Select Text layer for Text1/Text2/Text3  
                             #define TEXT1                    0x00000000  
                             #define TEXT2                    0x00000001  
                             #define TEXT3                    0x00000002  
nBlendValue:                Blend Value range is 0~63

Returns:                    (None)



---

## FadeIn( )

---

Prototype:                void FadeIn(void)

Descriptions:            To make fade in to color palette effect with hardware function

Arguments:              (None)

Returns:                  (None)

---

## FadeOut( )

---

Prototype:               void FadeOut(void)

Descriptions:            To make fade out to black screen effect with hardware function

Arguments:              (None)

Returns:                  (None)

Sunplus Confidential for SPECTRUM Use Only

## 9.19 PPU Register Set

### 0x8801\_0000 PPU Control

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1-D0
P_PPU_Control				PPUEN											Resolution

**Resolution:** Select Resolution Mode

- 00: QVGA(320x240)
- 01: VGA(640x480)
- 10: HVGA(640x240)
- 11: VGA to CIF(320x240)

**PPUEN:** PPU Enable

- 0: Disable
- 1: Enable

### 0x8801\_0004 Sprite Control Registers

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_SP_Control															Coord_sel	Sp_en

**Coord\_sel:** Sprite Coordinate Select

- 0: Origin on center
- 1: Origin on top/left corner

**Sp\_en:** Sprite Enable

- 0: Disable
- 1: Enable

### 0x8801\_0008 Sprite Max Num

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_SP_Max																Sp_Max

**Sp\_Max:** Sprite Max available sprite numbers

Range: 0~511

### 0x8801\_000C Blend Sub

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Blend_Sub																Blend_Sub

**Blend\_Sub:** Blend formula select

- 0:  $C=A*\alpha + B*(1-\alpha)$
- 1:  $C=A*\alpha - B*(1-\alpha)$



## 0x8801\_0010 Trans RGB

Name	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Trans_RGB	TransRGB_En	TransRGB															

**TransRGB:** Transparent RGB[5:6:5]

If (Pixel\_Value == P\_Trans\_RGB)

Pixel Value is transparent color

**TransRGB\_En:** Transparent RGB enable

0: Disable

1: Enable

## 0x8801\_0020 Text1 X Position

## 0x8801\_003C Text2 X Position

## 0x8801\_0058 Text3 X Position

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx1_X_position							X1									
P_Tx2_X_position							X2									
P_Tx3_X_position							X3									

**X:** Text Visible Area Horizontal Position

Range: -512~+511

## 0x8801\_0024 Text1 Y Position

## 0x8801\_0040 Text2 Y Position

## 0x8801\_005C Text3 Y Position

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx1_Y_position							Y1									
P_Tx2_Y_position							Y2									
P_Tx3_Y_position							Y3									

**Y:** Text Visible Area Vertical Position

Range: -256~+255

## 0x8801\_0028 Text1 Attribute

## 0x8801\_0044 Text2 Attribute

## 0x8801\_0060 Text3 Attribute

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx1_attribute		Depth1		Palette1					Vs1		Hs1		Flip1		Color1	
P_Tx2_attribute		Depth2		Palette2					Vs2		Hs2		Flip2		Color2	
P_Tx3_attribute		Depth3		Palette3					Vs3		Hs3		Flip3		Color3	





**Color:** Text Color Mode  
00: 2 bits / pixel (4 colors) 01: 4 bits / pixel (16 colors)  
10: 6 bits / pixel (64 colors) 11: 8 bits / pixel (256 colors)

**Flip:** Text Character Flip  
00: No Flip 01: Horizontal Flip  
10: Vertical Flip 11: Vertical and Horizontal Flip

**Hs:** Text Character Horizontal Size  
00: 8 pixels 01: 16 pixels  
10: 32 pixels 11: 64 pixels

**Vs:** Text Character Vertical Size  
00: 8 pixels 01: 16 pixels  
10: 32 pixels 11: 64 pixels

**Palette:** Text Palette Select  
Range: 0~63

**Depth:** Text Depth  
00: Depth 0 (bottom)  
01: Depth 1  
02: Depth 2  
03: Depth 3 (top)

**0x8801\_002C Text1 Control**

**0x8801\_0048 Text2 Control**

**0x8801\_0064 Text3 Control**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx1_control				RGB565				Blend1	RGB555	VCmp1	HCmp	Mve1	Txe1	Wap1	Rgm1	Linr1
P_Tx2_control				RGB565				Blend2	RGB555	VCmp2		Mve2	Txe2	Wap2	Rgm2	Linr2
P_Tx3_control				RGB565				Blend3	RGB555	VCmp3		Mve3	Txe3	Wap3	Rgm3	Linr3

**Linr:** Text Bitmap Mode Enable  
0: Character Mode  
1: Bitmap Mode

**Rgm:** Text Register Mode Enable  
0: Character Attribute Array Effective  
1: Character Register Set Effective

**Wap:** Text Wallpaper Effect Enable  
0: Disable (Normal)  
1: Enable (Only First Character/Line attribute is effective)

**Txe:** Text Enable

0: Disable (Non Visible)

1: Enable (Visible)

**Mve:** Text Horizontal Movement Control Enable

0: Disable

1: Enable

**HCmp:** Text Horizontal extension/compression enable.

**(Horizontal extension/compression is only available for Text1)**

0: Disable

1: Enable

**VCmp:** Text1 Vertical compression/extension enable

0: Disable

1: Enable

**RGB555:** Text 32768 Colors Mode( ARGB1555)

0: Disable

1: Enable

**Blend:** Text Blend control

0: Normal mode 1: Blending mode

**RGB565:** Text 65536 Colors Mode( RGB565)

0: Disable

1: Enable

**0x8801\_0030** Text1 Character number array start address or bitmap line start address

**0x8801\_004C** Text2 Character number array start address or bitmap line start address

**0x8801\_0068** Text3 Character number array start address or bitmap line start address

Name	D31 – D0
P_Tx1_N_PTR	Ptrnum1[31:0]
P_Tx2_N_PTR	Ptrnum2[31:0]
P_Tx3_N_PTR	Ptrnum3[31:0]

**Ptrnum:** Text Screen Number Array Page

**0x8801\_0038 Text1 Blend Level**

**0x8801\_0054 Text2 Blend Level**

**0x8801\_0070 Text3 Blend Level**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx1_Blend_Level											Tx1_Blend_Level					
P_Tx2_Blend_Level											Tx2_Blend_Level					
P_Tx3_Blend_Level											Tx3_Blend_Level					

**Tx\_Blend\_Level:** Text Blend Level

Range: 0 ~ 63

**0x8801\_0074 Vertical compression/extension scale register**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_VComp_Value											VCmpValue					

**VCmpValue:** Text Screen vertical compression/extension value.

Range: 0 ~ 1024

Double size: 10h.

Unit gain: 20h.

Half size: 40h

**0x8801\_0078 Vertical compression/extension scale movement control register**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_VComp_Offset				VCmpOffset												

**VCmpOffset:** Text Screen vertical compression/extension movement value.

**0x8801\_007C Vertical compression/extension scale movement control register**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_VComp_Step											VCmpStep					

**VCmpStep:** Text Screen vertical compression/extension step value.

Range: -128 ~ +127



## 0x8801\_0080 IRQ Control

Name	D15-D3	D2	D1	D0
P_IRQ_control		HV_IRQ_EN	BLK_END_EN	BLK_ST_EN

**BLK\_ST\_EN:** VBlanking Start IRQ

0: Disable 1: Enable

**BLK\_END\_EN:** VBlanking End IRQ

0: Disable 1: Enable

**HV\_IRQ\_EN:** Horizontal and vertical Position hit IRQ

0: Disable 1: Enable

## 0x8801\_0084 IRQ Status

Name	D15-D3	D2	D1	D0
P_IRQ_Status		HV_IRQ	BLK_IRQ_END	BLK_IRQ_ST

**BLK\_IRQ\_ST:** VBlanking Start IRQ status

0: None 1: Occurred

**BLK\_IRQ\_END:** VBlanking End IRQ status

0: None 1: Occurred

**HV\_IRQ:** Horizontal and vertical position hit IRQ status

0: None 1: Occurred

## 0x8801\_0088 Video Timing Vertical IRQ register

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_IRQTMV																

**IRQTimingV:** H/V position hit IRQ Vertical position

Range: 0 ~ 1023

## 0x8801\_0088 Video Timing Horizontal IRQ register

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_IRQTMH																

**IRQTimingH:** H/V position hit IRQ Horizontal position

Range: 0 ~ 1023

## 0x8801\_0090 Blanking Time

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_VBLK_TIME																

**VBlank\_Time:** Vertical blanking time extension

Range: 0~1023 lines

**0x8801\_00A0~0x8801\_00CC Buffer Start Address**

Name	D31-D0
P_TX1_BUF_SA1	Tx1_BUF_Start_Address1[31:0]
P_TX1_BUF_SA2	Tx1_BUF_Start_Address2[31:0]
P_TX1_BUF_SA3	Tx1_BUF_Start_Address3[31:0]
P_TX2_BUF_SA1	Tx2_BUF_Start_Address1[31:0]
P_TX2_BUF_SA2	Tx2_BUF_Start_Address2[31:0]
P_TX2_BUF_SA3	Tx2_BUF_Start_Address3[31:0]
P_TX3_BUF_SA1	Tx3_BUF_Start_Address1[31:0]
P_TX3_BUF_SA2	Tx3_BUF_Start_Address2[31:0]
P_TX3_BUF_SA3	Tx3_BUF_Start_Address3[31:0]
P_Frame_BUF_SA1	Frame_BUF_Start_Address1[31:0]
P_Frame_BUF_SA2	Frame_BUF_Start_Address2[31:0]
P_Frame_BUF_SA3	Frame_BUF_Start_Address3[31:0]
P_SP_BUF_SA	Sprite_BUF_Start_Address[31:0]

**P\_Tx\_BUF\_SA1:** Character data of Text start address / Bitmap buffer 1

\* When choosing character mode of TEXT, P\_Tx\_BUF\_SA1 will be the start address of character data in memory

**P\_Tx\_BUF\_SA2:** Bitmap buffer 2

**P\_Tx\_BUF\_SA3:** Bitmap buffer 3

**P\_Frame\_BUF\_SA:** PPU output frame buffer 1~3 start address

**P\_SP\_BUF\_SA:** Sprite data buffer start address

**0x8801\_1000~0x8801\_17FC Color Palette**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Color_Palet	Tr	Palette_R					Palette_G					Palette_B				

**Tr:** Transparent color control

0: Normal

1: Transparent

**Palette\_R:** Color Red component

**Palette\_G:** Color Green component

**Palette\_B:** Color Blue component

**0x8801\_2000 ~ 0x8801\_27FC Text Horizontal movement control registers**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx_Hvoffset							Tx_Line_Offset_Value									

**Hvoffset0:** Text Screen line0 offset

**Hvoffset1:** Text Screen line1 offset



...

**Hvoffset511:** Text Screen line511 offset

**0x8801\_3000 ~ 0x8801\_37FC Horizontal extension/compression scale registers**

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_HComp_Value									Tx_Line_HCmp_Value							

**HCmpValue0:** Text Screen line0 extension/ compression value

**HCmpValue 1:** Text Screen line1 extension/ compression value

...

**HCmpValue 511:** Text Screen line511 extension/ compression value.

Unit gain: 80h

Half size: 40h

**0x8801\_4000 ~ 0x8801\_4FFE Sprite Attributes**
**0x8801\_4000** P\_Sprite0\_Num\_X

**0x8801\_4004** P\_Sprite0\_Att\_Y

**0x8801\_4008** P\_Sprite1\_Num\_X

**0x8801\_400C** P\_Sprite1\_Att\_Y

:

:

Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Sprite_Ch_Num	Sprite_Character_number															
Name	D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
P_Sprite_X							Sprite_X									
Name	D15		D14	D13	D12-D8			D7	D6	D5	D4	D3	D2	D1	D0	
P_Sprite_Attribute	Sprite_Blend		Sprite_Depth		Sprite_Palette			Sprite_vsize		Sprite_hsize		Sprite_Flip		Sprite_Color		
Name	D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
P_Sprite_Y	Sprite_Blend_Level						Sprite_Y									

**Sprite\_Character\_Number:** Sprite Character Number

Range: 0~ 65535

**Sprite\_X:**

Sprite Horizontal Position

Range: 0~ 1023

**Sprite\_Blend:**

Blend control

0: Disable      1: Enable

**Sprite\_Paltte:**

Color Palette Select

Range: 0~63

**Sprite\_Depth:**

Sprite Depth

00:      Depth 0 (bottom)



	01:	Depth 1
	02:	Depth 2
	03:	Depth 3 (top)
<b>Sprite_Vsize:</b>	Character Vertical Size	
	00: 8 pixels	01: 16 pixels
	10: 32 pixels	11: 64 pixels
<b>Sprite_Hsize:</b>	Character Horizontal Size	
	00: 8 pixels	01: 16 pixels
	10: 32 pixels	11: 64 pixels
<b>Sprite_Flip:</b>	Character Flip mode	
	00: No Flip	01: Horizontal Flip
	10: Vertical Flip	11: Vertical and Horizontal Flip
<b>Sprite_Color :</b>	Color Mode	
	00: 2 bits / pixel	01: 4 bits / pixel
	10: 6 bits / pixel	11: 8 bits / pixel
<b>Sprite_Blend_Level:</b>	Sprite blend level	
	Range: 0~ 63	
<b>Sprite_Y:</b>	Vertical Position	
	Range: 0~ 1023	

Name	Port 8801_XXXX	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
P_PPU_Control	0000				PPUEN											Resolution		
P_SP_Control	0004															Coord_s el	Sp_en	
P_SP_Max	0008								Sp_Max									
P_Blend_Sub	000C																	Blend_Sub
P_Trans_RGB	0010	TransRGB_En[16]/Trans_RGB[15:0]																
Reserved	0014	Reserved																
Reserved	0018	Reserved																
Reserved	001C	Reserved																
P_Tx1_X_position	0020								X1									
P_Tx1_Y_position	0024									Y1								
P_Tx1_attribute	0028		Depth1		Palette1					Vs1		Hs1		Flip1		Color1		
P_Tx1_control	002C				RGB565				Blend1	RGB555	VCmp1	HCmp	Mve1	Txe1	Wap1	Rgm1	Linr1	
P_Tx1_N_PTR	0030	Ptrnum1[31:0]																
Reserved	0034	Reserved																
P_Tx1_Blend_Level	0038									Tx1_Blend_Level								
P_Tx2_X_position	003C								X2									
P_Tx2_Y_position	0040									Y2								
P_Tx2_attribute	0044		Depth2		Palette2					Vs2		Hs2		Flip2		Color2		
P_Tx2_control	0048				RGB565				Blend2	RGB555	VCmp2		Mve2	Txe2	Wap2	Rgm2	Linr2	
P_Tx2_N_PTR	004C	Ptrnum2[31:0]																
Reserved	0050	Reserved																



Name	Port 8801_XXXX	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
P_Tx2_Blend_Level	0054									Tx2_Blend_Level								
P_Tx3_X_position	0058								X3									
P_Tx3_Y_position	005C									Y3								
P_Tx3_attribute	0060		Depth3		Palette3					Vs3		Hs3		Flip3		Color3		
P_Tx3_control	0064				RGB565				Blend3	RGB555	VCmp3		Mve3	Txe3	Wap3	Rgm3	Linr3	
P_Tx3_N_PTR	0068	Ptrnum3[31:0]																
Reserved	006C	Reserved																
P_Tx3_Blend_Level	0070									Tx3_Blend_Level								
P_VComp_Value	0074								VCmpValue									
P_VComp_Offset	0078				VCmpOffset													
P_VComp_Step	007C									VCmpStep								
P_IRQ_control	0080													HV_IRQ_EN	BLK_END_EN	BLK_ST_EN		
P_IRQ_Status	0084													HV_IRQ_Q	BLK_IRQ_END	BLK_IRQ_ST		
P_IRQTMV	0088								IRQTimingV									
P_IRQTMH	008C								IRQTimingH									
P_VBLK_TIME	0090								Vertical Blank Time (unit: lines)									
P_TX1_BUF_SA0	00A0	Tx1_BUF_Start_Address0[31:0]																
P_TX1_BUF_SA1	00A4	Tx1_BUF_Start_Address1[31:0]																
P_TX1_BUF_SA2	00A8	Tx1_BUF_Start_Address2[31:0]																
P_TX2_BUF_SA0	00AC	Tx2_BUF_Start_Address0[31:0]																

Name	Port 8801_XXXX	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_TX2_BUF_SA1	00B0	Tx2_BUF_Start_Address1[31:0]															
P_TX2_BUF_SA2	00B4	Tx2_BUF_Start_Address2[31:0]															
P_TX3_BUF_SA0	00B8	Tx3_BUF_Start_Address0[31:0]															
P_TX3_BUF_SA1	00BC	Tx3_BUF_Start_Address1[31:0]															
P_TX3_BUF_SA2	00C0	Tx3_BUF_Start_Address2[31:0]															
P_Frame_BUF_SA0	00C4	Frame_BUF_Start_Address0[31:0]															
P_Frame_BUF_SA1	00C8	Frame_BUF_Start_Address1[31:0]															
P_Frame_BUF_SA2	00CC	Frame_BUF_Start_Address1[31:0]															
P_SP_BUF_SA	00D0	Sprite_BUF_Start_Address[31:0]															
P_Color_Palet0	1000	Tr0	Palette_R_0					Palette_G_0					Palette_B_0				
P_Color_Palet1	1004	Tr0	Palette_R_1					Palette_G_1					Palette_B_1				
:	:	:	:					:					:				
P_Color_Palet510	17F8	Tr510	Palette_R_510					Palette_G_510					Palette_B_510				
P_Color_Palet511	17FC	Tr511	Palette_R_511					Palette_G_511					Palette_B_511				
P_Tx_Hvoffset0	2000							Tx_Line0_Hoffset_Value									
P_Tx_Hvoffset1	2004							Tx_Line1_Hoffset_Value									
....	....	....						....									

Name	Port 8801_XXXX	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Tx_Hvoffset510	27F8							Tx_Line510_Hoffset_Value									
P_Tx_Hvoffset511	27FC							Tx_Line511_Hoffset_Value									
Horizontal Compression Control Registers																	
P_HComp_Value0	3000							Tx_Line0_HCmp_Value									
P_HComp_Value1	3004							Tx_Line1_HCmp_Value									
:	:	:						:									
P_HComp_Value510	37F8							Tx_Line510_HCmp_Value									
P_HComp_Value511	37FC							Tx_Line511_HCmp_Value									
Sprite Control Registers																	
P_Sprite0_Ch_Num	4000	Sprite0_Character_number															
P_Sprite0_X	4002							Sprite0_X									
P_Sprite0_Attribute	4004	Sprite0_Blend	Sprite0_Depth		Sprite0_Palette				Sprite0_vsize		Sprite0_hsize		Sprite0_Flip		Sprite0_Color		
P_Sprite0_Y	4006	Sprite0_Blend_Level						Sprite0_Y									
P_Sprite1_Ch_Num	4008	Sprite1_Character_number															
P_Sprite1_X	400A							Sprite1_X									
P_Sprite1_Attribute	400C	Sprite1_Blend	Sprite1_Depth		Sprite1_Palette				Sprite1_vsize		Sprite1_hsize		Sprite1_Flip		Sprite1_Color		
P_Sprite1_Y	400E	Sprite1_Blend_Level						Sprite1_Y									

Name	Port 8801_XXXX	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
P_Sprite511_Ch_Nu m	4FF8	Sprite511_Character_number															
P_Sprite511_X	4FFA									Sprite511_X							
P_Sprite511_Attribut e	4FFC	Sprite511_Blend	Sprite511_Depth		Sprite511_Palette					Sprite511_vsize		Sprite511_hsize		Sprite511_Flip		Sprite511_Color	
P_Sprite511_Y	4FFE	Sprite511_Blend_Level						Sprite511_Y									

---

## 10 BITBLT DMA WITH ALPHA BLENDING OPERATION

---

### 10.1 Features

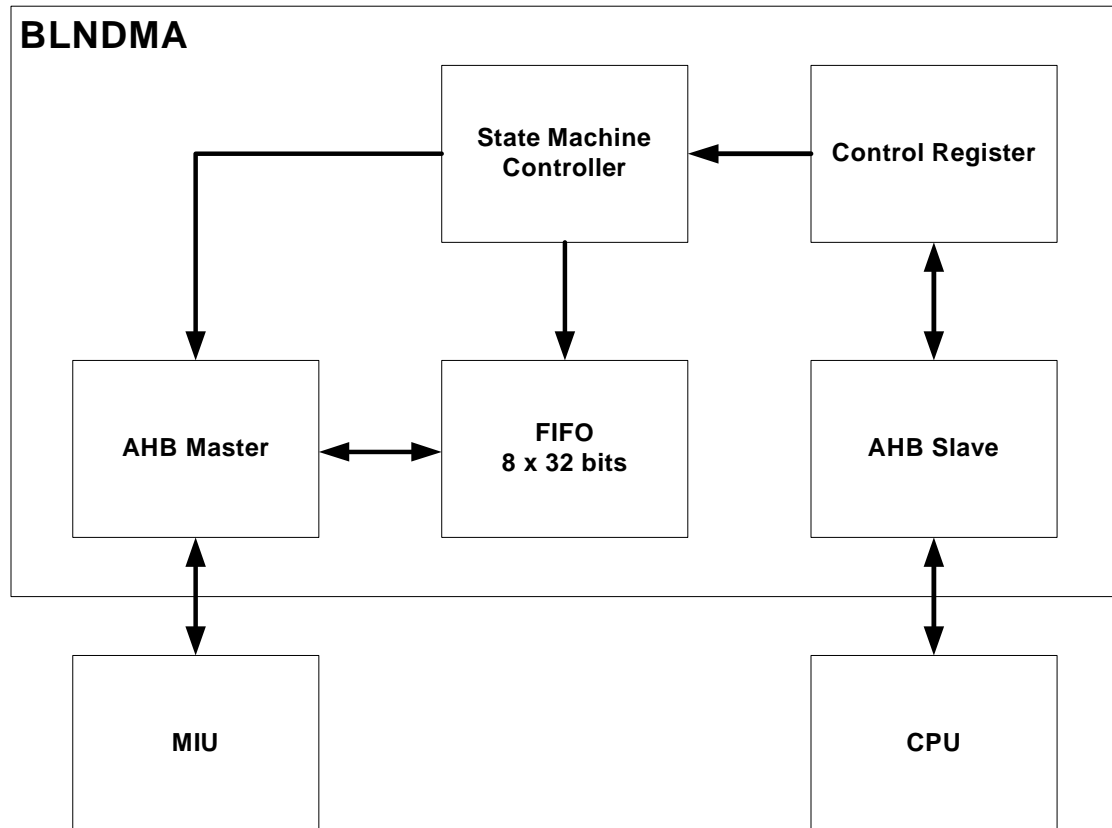
The BLNDMA Controller supports a DRAM-to-DRAM DMA function. This function performs data transfer from source address to destination address in the SDRAM. Source address, destination address and transfer word count must be programmed before the transfer. The transfer is word alignment. One word includes two pixel data. There are the three main functions for the BLNDMA Controller. One is the DMA Pattern Fill function, the second is the DRAM-to-DRAM DMA function, the third is the DRAM-to-DRAM DMA with the transparent filter function.

If it performs the DMA Pattern Fill Function, Pattern is decided by the 32 bit Register "DMA\_FILL\_PAT" for the register address 0x080D\_0010. If it performs the DRAM-to-DRAM DMA function, the transfer word count is decided by the "DMA\_TRANS\_WIDTH" and "DMA\_TRANS\_HEIGHT" for the register address 0x080D\_000C. They are 12 bit for each other, that is, the maximal transfer size is 4096 x 4096 words. The "DMA\_TRANS\_WIDTH" must be the multiple time of the 8 words. The BLNDMA Controller only supports the linear address mode and not supports the block address mode. If it performs the DRAM-to-DRAM DMA with the transparent filter function, the transparent filter pattern is decided by the 16 bit Register "DMA\_FILTER\_PAT" for the register address 0x080D\_0020. If the transferred data is matched with the transparent filter pattern, then it does not write into the destination address in the SDRAM.

When the BLNDMA Controller performs the above functions, there are the two way to let CPU know the BLNDMA status. One is the Polling Mode. CPU can read the bit 8 "DMA\_STATUS" for the register address 0x080D\_0018 and monitors the BLNDMA Controller whether the data transfer completes or not. The second is the Interrupt Mode. When the data transfer completes, the BLNDMA Controller sends the interrupt to the CPU.

## 10.2 Architecture

The architecture of the BLNDMA Controller is shown below.



The BLNDMA Controller is consisted of the AHB Slave, State Machine Controller and AHB Master. CPU can access the internal control register of the BLNDMA controller by the AHB Slave of the BLNDMA controller. It can activate the state machine controller by the control register setting. Then the state machine controller controls the AHB Master of the BLNDMA controller and access the data in the SDRAM through the MIU controller. For each time the AHB Master reads 8 words data from MIU into the internal FIFO of the BLNDMA controller and performs the function of the BLNDMA controller, then writes into the SDRAM.



### 10.3 Control registers

#### 10.3.1 P\_BLNDMA\_SRC\_A\_ADDR(R/W)(0x880D0000): DMA SourceA Starting Address

NAME	D27-D0
P_BLNDMA_SRC_A_ADDR	DMA_SRC_A_ADDR

#### 10.3.2 P\_BLNDMA\_SRC\_B\_ADDR(R/W)(0x880D0004): DMA SourceB Starting Address

NAME	D27-D0
P_BLNDMA_SRC_B_ADDR	DMA_SRC_B_ADDR

#### 10.3.3 P\_BLNDMA\_DEST\_ADDR(R/W)(0x880D0008): DMA Destination Starting Address

NAME	D27-D0
P_BLNDMA_DEST_ADDR	DMA_DEST_ADDR

#### 10.3.4 P\_BLNDMA\_WIDTH\_HEIGHT(R/W)(0x880D000C): DMA Transfer Width

NAME	D27-D16	D11-D0
P_BLNDMA_WIDTH_HEIGHT	DMA_TRANS_HEIGHT	DMA_TRANS_WIDTH

Note: 1. Unit is word, one word including two pixel data

2. Must be the multiple time of the 8 words

#### 10.3.5 P\_BLNDMA\_FILL\_PAT(R/W)(0x880D0010): DMA Fill Pattern

NAME	D31-D0
P_BLNDMA_FILL_PAT	DMA_FILL_PAT

#### 10.3.6 P\_BLNDMA\_CONTROL\_1(R/W)(0x880D0014): DMA Operation Mode

NAME	D24	D23-D17	D16	D15-D9	D8	D7-D2	D1	D0
P_BLNDMA_CONTROL_1	DMA_START		FILTER_MODE		BLEND_EN		OP_MODE	

OP\_Mode: DMA operation mode

- 0: Idle mode
- 1: Copy A and Paste to Destination
- 2: Reserved and prohibition usage
- 3: Fill Pattern to Destination

BLEND\_EN: Blending Effect enable

- 0: Disable
- 1: Enable

Filter\_Mode: DMA transparent filter mode

- 0: Disable
- 1: Enable

DMA\_START: DMA operating function start

- 0: Disable



1: Enable

#### 10.3.7 P\_BLNDMA\_IRQ\_CONTROL(R/W)(0x880D0018): DMA Fill Pattern

NAME	D24	D23-D17	D16	D15-D9	D8	D7-D0
P_BLNDMA_IRQ_CONTROL	DMA_INT_CLEAR		DMA_INT_EN		DMA_STATUS	

DMA\_STATUS: DMA operation status

0: DMA Operation is Done

1: DMA Operation is Busy

DMA\_INT\_EN: DMA Interrupt Control

0: Disable

1: Enable

DMA\_INT\_CLEAR: DMA Interrupt Clear

0: none

1: Clear DMA Interrupt

#### 10.3.8 P\_BLNDMA\_BLEND\_FACTOR(R/W)(0x880D001C):

NAME	D13-D8	D7-D6	D5-D-
P_BLNDMA_BLEND_FACTOR	DMA_BLEND_FACB		DMA_BLEND_FACA

DMA\_BLEND\_FACA: DMA Blend Factor A

DMA\_BLEND\_FACB: DMA Blend Factor B

#### 10.3.9 P\_BLNDMA\_TRANSPARENT(R/W)(0x880D0020):

NAME	D15-D0
P_BLNDMA_TRANSPARENT	COLOR_KEY

#### 10.3.10 P\_BLNDMA\_ADDR\_MODE(R/W)(0x880D0024):

NAME	D31-D0
P_BLNDMA_ADDR_MODE	DMA_ADDR_MODE

#### 10.3.11 P\_BLNDMA\_CONTROL\_2(R/W)(0x880D0028):

NAME	D8	D7-D1	D0
P_BLNDMA_CONTROL_2	DMA_COLOR_MODE		DMA_ALPHA

DMA\_ALPHA: Alpha select

0: Disable

1: Enable

DMA\_COLOR\_MODE: RGB type select

0: RGB565





1: ARGB1555

**10.3.12 P\_BLNDMA\_ABASE\_ADDR(R/W)(0x880D0030):**

NAME	D27-D0
P_BLNDMA_ABASE_ADDR	DMA_ABASE_ADDR

**10.3.13 P\_BLNDMA\_AOFFSET\_XY(R/W)(0x880D0034):**

NAME	D26-D16	D15-D11	D10-D0
P_BLNDMA_AOFFSET_XY	OFFSETA_Y		OFFSETA_X

OFFSETA\_Y/OFFSETA\_X: Maximum = 2048

**10.3.14 P\_BLNDMA\_A\_BG(R/W)(0x880D0038):**

NAME	D10-D8	D7-D3	D2-D0
P_BLNDMA_A_BG	BG_HEIGH		BG_WIDTH

BG\_HEIGH/BG\_WIDTH: Background Heigh & Width

- 0: 256
- 1: 320
- 2: 512
- 3: 640
- 4: 1024
- 5: 2048

**10.3.15 P\_BLNDMA\_BBASE\_ADDR(R/W)(0x880D0040):**

NAME	D27-D0
P_BLNDMA_BBASE_ADDR	DMA_BBASE_ADDR

**10.3.16 P\_BLNDMA\_BOFFSET\_XY(R/W)(0x880D0044):**

NAME	D26-D16	D15-D11	D10-D0
P_BLNDMA_BOFFSET_XY	OFFSETB_Y		OFFSETB_X

OFFSETB\_Y/OFFSETB\_X: Maximum = 2048

**10.3.17 P\_BLNDMA\_B\_BG(R/W)(0x880D0048):**

NAME	D10-D8	D7-D3	D2-D0
P_BLNDMA_B_BG	BG_HEIGH		BG_WIDTH

BG\_HEIGH/BG\_WIDTH: Background Heigh & Width

- 0: 256
- 1: 320



2: 512  
3: 640  
4: 1024  
5: 2048

**10.3.18 P\_BLNDMA\_DBASE\_ADDR(R/W)(0x880D0050):**

NAME	D27-D0
P_BLNDMA_DBASE_ADDR	DMA_DBASE_ADDR

**10.3.19 P\_BLNDMA\_DOFFSET\_XY(R/W)(0x880D0054):**

NAME	D26-D16	D15-D11	D10-D0
P_BLNDMA_DOFFSET_XY	OFFSETD_Y		OFFSETD_X

OFFSETD\_Y/OFFSETD\_X: Maximum = 2048

**10.3.20 P\_BLNDMA\_D\_BG(R/W)(0x880D0058):**

NAME	D10-D8	D7-D3	D2-D0
P_BLNDMA_D_BG	BG_HEIGH		BG_WIDTH

BG\_HEIGH/BG\_WIDTH: Background Heigh & Width

0: 256  
1: 320  
2: 512  
3: 640  
4: 1024  
5: 2048



#### 10.4 BLNDMA API Services

BInDmaStart\_LinearFillPat( )  
BInDmaStart\_BlockFillPat( )  
BInDmaStart\_Linear2Linear\_Dma( )  
BInDmaStart\_Linear2Block\_Dma( )  
BInDmaStart\_Block2Block\_Dma( )  
BInDmaStart\_Block2Linear\_Dma( )  
BInDmaStart\_Linear2Linear\_YCbCr2RGB( )  
BInDmaStart\_Linear2Block\_YCbCr2RGB( )  
BInDmaStart\_Block2Linear\_YCbCr2RGB( )  
BInDmaStart\_Block2Block\_YCbCr2RGB( )  
BInDmaStart\_Linear2Linear\_Blend( )  
BInDmaStart\_Linear2Block\_Blend( )  
BInDmaStart\_Block2Linear\_Blend( )  
BInDmaStart\_Block2Block\_Blend( )  
BInDmaStart\_Linear2Linear\_BlendwithDest( )  
BInDmaStart\_Linear2Block\_BlendwithDest( )  
BInDmaStart\_Block2Linear\_BlendwithDest( )  
BInDmaStart\_Block2Block\_BlendwithDest( )



AUG 25, 2005



AUG 25, 2005



AUG 25, 2005



AUG 25, 2005



### BInDmaStart\_Block2Block\_Dma( )

```

void BlnDmaStart_Block2Block_Dma(
    UINT32 src_base_addr,
    UINT32 src_offset_x,
    UINT32 src_offset_y,
    UINT8  src_bg_width,
    UINT8  src_bg_heigh,
    UINT32 dest_base_addr,
    UINT32 dest_offset_x,
    UINT32 dest_offset_y,
    UINT8  dest_bg_width,
    UINT8  dest_bg_heigh,
    UINT32 trans_width,
    UINT32 trans_heigh,
    UINT8  transparent_mode,
    UINT32 color_key,
    UINT8  mode )

```

Descriptions: Block Fill to Block

Arguments:

src_base_addr:	source A base address (28 bits)		
src_offset_x/dest_offset_x:	11 bits, maximum = 2048		
src_offset_y/dest_offset_y:	11 bits, maximum = 2048		
dest_base_addr:	destination base address (28 bits)		
trans_width:	transfer width, maximum = 2048		
trans_heigh:	transfer heigh, maximum = 2048		
src_bg_width/dest_bg_width:	dest/source background width(3 bits)		
	0 : 256	1 : 320	2 : 512
	3 : 640	4 : 1024	5 : 2048
src_bg_heigh/dest_bg_heigh:	dest/source background heigh(3 bits)		
	0 : 240	1 : 256	2 : 480
	3 : 512	4 : 1024	5 : 2048
transparent_mode:	DMA transparent filter mode		
	0: disable		1: enable
color_key:	Transparent color ARGB1555/RGB565		
mode:	Interrupt mode enable		
	0: polling mode		1: interrupt mode

Returns: (None)





Descriptions: Linear Fill to Block

src_base_addr:	source base address (28 bits)	
src_offset_x:	11 bits, maximum = 2048	
src_offset_y:	11 bits, maximum = 2048	
dest_addr:	destination address (28 bits)	
trans_width:	transfer width, maximum = 2048	
trans_heigh:	transfer heigh, maximum = 2048	
dest_bg_width:	dest background width(3 bits)	
	0 : 256	1 : 320
	2 : 512	3 : 640
	4 : 1024	5 : 2048
dest_bg_heigh:	dest background heigh(3 bits)	
	0 : 240	1 : 256
	2 : 480	3 : 512
	4 : 1024	5 : 2048
transparent_mode:	DMA transparent filter mode	
	0: disable	1: enable
color_key:	Transparent color ARGB1555/RGB565	
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

Returns: (None)



Prototype:	void BlnDmaStart_linear2linear_YCbCr2RGB(	UINT32 src_addr,
		UINT32 dest_addr,
		UINT32 trans_width,
		UINT32 trans_heigh,
		UINT8 color_mode,
		UINT8 alpha,
		UINT8 mode )

Arguments:

src_addr:	source address (28 bits)	
dest_addr:	destination address (28 bits)	
trans_width:	transfer width, maximum = 2048	
trans_heigh:	transfer heigh, maximum = 2048	
color_mode:	color mode	
	0: RGB565	1: ARGB1555
alpha:	alpha enable	
	0: Disable	1: enable
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

Returns: (None)



Descriptions: Linear Fill to Block

src_addr:	source address (28 bits)	
dest_base_addr:	destination address (28 bits)	
dest_offset_x:	11 bits, maximum = 2048	
dest_offset_y:	11 bits, maximum = 2048	
trans_width:	transfer width, maximum = 2048	
trans_heigh:	transfer heigh, maximum = 2048	
dest_bg_width:	dest background width(3 bits)	
	0 : 256	1 : 320
	2 : 512	3 : 640
	4 : 1024	5 : 2048
dest_bg_heigh:	dest background heigh(3 bits)	
	0 : 240	1 : 256
	2 : 480	3 : 512
	4 : 1024	5 : 2048
color_mode:	color mode	
	0: RGB565	1: ARGB1555
alpha:	alpha enable	
	0: Disable	1: enable
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

© Sunplus Technology Co., Ltd.



Descriptions: Linear Fill to Block

src_base_addr:	source base address (28 bits)	
src_offset_x:	11 bits, maximum = 2048	
src_offset_y:	11 bits, maximum = 2048	
src_bg_width:	source background width(3 bits)	
	0 : 256	1 : 320
	2 : 512	3 : 640
	4 : 1024	5 : 2048
src_bg_heigh:	source background heigh(3 bits)	
	0 : 240	1 : 256
	2 : 480	3 : 512
	4 : 1024	5 : 2048
dest_addr:	destination address (28 bits)	
trans_width:	transfer width, maximum = 2048	
trans_heigh:	transfer heigh, maximum = 2048	
color_mode:	color mode	
	0: RGB565	1: ARGB1555
alpha:	alpha enable	
	0: Disable	1: enable
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

© Sunplus Technology Co., Ltd.



Returns: (None)



### BlnDmaStart\_Linear2Linear\_Blend ( )

```

void BlnDmaStart_linear2linear_Blend(
    UINT32 srca_addr,
    UINT32 srcb_addr,
    UINT32 dest_addr,
    UINT32 trans_width,
    UINT32 trans_heigh,
    UINT8  blend_mode,
    UINT8  blend_a_factor,
    UINT8  blend_b_factor,
    UINT8  color_mode,
    UINT8  alpha,
    UINT8  transparent_mode,
    UINT32 color_key,
    UINT8  mode )

```

Descriptions: Linear Fill to Block

Arguments:

src_a_addr/src_b_addr/dest_addr:	sourceA/sourceB/destination base address (28 bits)	
trans_width:	transfer width, maximum = 2048	
trans_heigh:	transfer heigh, maximum = 2048	
blend_mode:	blend mode enable	
blend_a_factor:	blend a factor value(6 bits)	
blend_b_factor:	blend b factor value(6 bits)	
color_mode:	color mode	
	0: RGB565	1: ARGB1555
alpha:	alpha enable	
	0: Disable	1: enable
transparent_mode:	DMA transparent filter mode	
	0: disable	1: enable
color_key:	Transparent color ARGB1555/RGB565	
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

Returns: (None)



AUG 25, 2005



	0: Disable	1: enable
transparent_mode:	DMA transparent filter mode	
	0: disable	1: enable
color_key:	Transparent color ARGB1555/RGB565	
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

Returns: (None)





src_a_addr/src_b_addr/dest_addr:	sourceA/sourceB/dest base address(28 bits)		
src_a_offset_x/src_b_offset_x:	11 bits, maximum = 2048		
src_a_offset_y/src_b_offset_y:	11 bits, maximum = 2048		
src_a_bg_width/src_b_bg_width:	destination background width(3 bits)		
	0 : 256	1 : 320	2 : 512
	3 : 640	4 : 1024	5 : 2048
src_a_bg_height/src_b_bg_height:	destination background height(3 bits)		
	0 : 240	1 : 256	2 : 480
	3 : 512	4 : 1024	5 : 2048
trans_width/trans_height:	transfer width/height, maximum = 2048		
blend_mode:	blend mode enable		



blend_a_factor/blend_b_factor:	blend a/b factor value(6 bits)
color_mode:	color mode
	0: RGB565                      1: ARGB1555
alpha:	alpha enable
	0: Disable                      1: enable
transparent_mode:	DMA transparent filter mode
	0: disable                      1: enable
color_key:	Transparent color ARGB1555/RGB565
mode:	Interrupt mode enable
	0: polling mode                      1: interrupt mode

Returns:                      (None)



src_a_addr/src_b_addr/dest_addr:	sourceA/sourceB/dest base address(28 bits)		
src_a_offset_x/src_b_offset_x/dest_offset_x:	11 bits, maximum = 2048		
src_a_offset_y/src_b_offset_y/dest_offset_y:	11 bits, maximum = 2048		
src_a_bg_width/src_b_bg_width/dest_bg_width:	destination background width(3 bits)		
	0 : 256	1 : 320	2 : 512
	3 : 640	4 : 1024	5 : 2048
src_a_bg_height/src_b_bg_height/dest_bg_height:	destination background height(3 bits)		



	0 : 240	1 : 256	2 : 480
	3 : 512	4 : 1024	5 : 2048
trans_width/trans_heigh:	transfer width/heigh, maximum = 2048		
blend_mode:	blend mode enable		
blend_a_factor/blend_b_factor:	blend a/b factor value(6 bits)		
color_mode:	color mode		
	0: RGB565	1: ARGB1555	
alpha:	alpha enable		
	0: Disable	1: enable	
transparent_mode:	DMA transparent filter mode		
	0: disable	1: enable	
color_key:	Transparent color ARGB1555/RGB565		
mode:	Interrupt mode enable		
	0: polling mode	1: interrupt mode	

Returns: (None)



### BInDmaStart\_Linear2Linear\_BlendwithDest ( )

```

void BlnDmaStart_linear2linear_BlendwithDest(
    UINT32 src_addr,
    UINT32 dest_addr,
    UINT32 trans_width,
    UINT32 trans_heigh,
    UINT8  blend_mode,
    UINT8  blend_a_factor,
    UINT8  blend_b_factor,
    UINT8  color_mode,
    UINT8  alpha,
    UINT8  transparent_mode,
    UINT32 color_key,
    UINT8  mode )

```

Descriptions: Linear Fill to Block

Arguments:

src_addr/dest_addr:	source/dest base address(28 bits)	
trans_width/trans_heigh:	transfer width/heigh, maximum = 2048	
blend_mode:	blend mode enable	
blend_a_factor/blend_b_factor:	blend a/b factor value(6 bits)	
color_mode:	color mode	
	0: RGB565	1: ARGB1555
alpha:	alpha enable	
	0: Disable	1: enable
transparent_mode:	DMA transparent filter mode	
	0: disable	1: enable
color_key:	Transparent color ARGB1555/RGB565	
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode

Returns: (None)



### BInDmaStart\_Linear2Block\_BlendwithDest ( )

```

void BlnDmaStart_linear2block_BlendwithDest(
    uint32_t src_addr,
    uint32_t dest_addr,
    uint32_t dest_offset_x,
    uint32_t dest_offset_y,
    uint8_t dest_bg_width,
    uint8_t dest_bg_height,
    uint32_t trans_width,
    uint32_t trans_height,
    uint8_t blend_mode,
    uint8_t blend_a_factor,
    uint8_t blend_b_factor,
    uint8_t color_mode,
    uint8_t alpha,
    uint8_t transparent_mode,
    uint32_t color_key,
    uint8_t mode )

```

Descriptions: Linear Fill to Block

Arguments:

src_addr/dest_addr:	source/dest base address(28 bits)		
dest_offset_x:	11 bits, maximum = 2048		
dest_offset_y:	11 bits, maximum = 2048		
dest_bg_width:	destination background width(3 bits)		
	0 : 256	1 : 320	2 : 512
	3 : 640	4 : 1024	5 : 2048
dest_bg_heigh:	destination background heigh(3 bits)		
	0 : 240	1 : 256	2 : 480
	3 : 512	4 : 1024	5 : 2048
trans_width/trans_heigh:	transfer width/heigh, maximum = 2048		
blend_mode:	blend mode enable		
blend_a_factor/blend_b_factor:	blend a/b factor value(6 bits)		
color_mode:	color mode		
	0: RGB565	1: ARGB1555	
alpha:	alpha enable		
	0: Disable	1: enable	



transparent_mode:	DMA transparent filter mode
	0: disable                      1: enable
color_key:	Transparent color ARGB1555/RGB565
mode:	Interrupt mode enable
	0: polling mode              1: interrupt mode

Returns:              (None)



### BInDmaStart\_Block2Linear\_BlendwithDest ( )

```

void BlnDmaStart_block2linear_BlendwithDest(
    UINT32 src_addr,
    UINT32 src_offset_x,
    UINT32 src_offset_y,
    UINT8  src_bg_width,
    UINT8  src_bg_heigh,
    UINT32 dest_addr,
    UINT32 trans_width,
    UINT32 trans_heigh,
    UINT8  blend_mode,
    UINT8  blend_a_factor,
    UINT8  blend_b_factor,
    UINT8  color_mode,
    UINT8  alpha,
    UINT8  transparent_mode,
    UINT32 color_key,
    UINT8  mode )

```

Descriptions: Linear Fill to Block

Arguments:

src_addr/dest_addr:	source/dest base address(28 bits)		
src_offset_x:	11 bits, maximum = 2048		
src_offset_y:	11 bits, maximum = 2048		
src_bg_width:	source background width(3 bits)		
	0 : 256	1 : 320	2 : 512
	3 : 640	4 : 1024	5 : 2048
src_bg_heigh:	source background heigh(3 bits)		
	0 : 240	1 : 256	2 : 480
	3 : 512	4 : 1024	5 : 2048
trans_width/trans_heigh:	transfer width/heigh, maximum = 2048		
blend_mode:	blend mode enable		
blend_a_factor/blend_b_factor:	blend a/b factor value(6 bits)		
color_mode:	color mode		
	0: RGB565	1: ARGB1555	
alpha:	alpha enable		
	0: Disable	1: enable	





transparent_mode:	DMA transparent filter mode
	0: disable                      1: enable
color_key:	Transparent color ARGB1555/RGB565
mode:	Interrupt mode enable
	0: polling mode                1: interrupt mode

Returns:                (None)



### BInDmaStart\_Block2Linear\_BlendwithDest ( )

```

void BlnDmaStart_block2block_BlendwithDest(
    UINT32 src_addr,
    UINT32 src_offset_x,
    UINT32 src_offset_y,
    UINT8  src_bg_width,
    UINT8  src_bg_heigh,
    UINT32 dest_addr,
    UINT32 dest_offset_x,
    UINT32 dest_offset_y,
    UINT8  dest_bg_width,
    UINT8  dest_bg_heigh,
    UINT32 trans_width,
    UINT32 trans_heigh,
    UINT8  blend_mode,
    UINT8  blend_a_factor,
    UINT8  blend_b_factor,
    UINT8  color_mode,
    UINT8  alpha,
    UINT8  transparent_mode,
    UINT32 color_key,
    UINT8  mode )

```

Descriptions: Linear Fill to Block

Arguments:

src_addr/dest_addr:	source/dest base address(28 bits)		
src_offset_x/dest_offset_x:	11 bits, maximum = 2048		
src_offset_y/dest_offset_y:	11 bits, maximum = 2048		
src_bg_width/dest_bg_width:	source/dest background width(3 bits)		
	0 : 256	1 : 320	2 : 512
	3 : 640	4 : 1024	5 : 2048
src_bg_heigh/dest_bg_heigh:	source/dest background heigh(3 bits)		
	0 : 240	1 : 256	2 : 480
	3 : 512	4 : 1024	5 : 2048
trans_width/trans_heigh:	transfer width/heigh, maximum = 2048		
blend_mode:	blend mode enable		
blend_a_factor/blend_b_factor:	blend a/b factor value(6 bits)		



color_mode:	color mode	
	0: RGB565	1: ARGB1555
alpha:	alpha enable	
	0: Disable	1: enable
transparent_mode:	DMA transparent filter mode	
	0: disable	1: enable
color_key:	Transparent color ARGB1555/RGB565	
mode:	Interrupt mode enable	
	0: polling mode	1: interrupt mode
Returns:	(None)	

## 11 DISPLAY UNIT - TVE

### 11.1 Description

The display unit, TV encoder unit in SPG290 is a multi-TV system and multi-screen mode TV encoder with 9 bits video DAC to generate composite video signal to TV screen with VGA resolution.

### 11.2 TV System and Screen Mode

The display unit can support several TV systems with different screen modes as the list shown below.

NTSC		PAL	
Interlace	Non-Interlace	Interlace	Non-Interlace
640 x 480	640 x 240	640 x 480	640 x 240
320 x 240	320 x 240	320 x 240	320 x 240

\* PAL system includes PAL-(B,D,G,H,I,N) modes

### 11.3 Frame Buffer

In SPG290, frame buffer can be rendered by PPU, CPU or MP4 decoder and with intelligent **Buffer Control Unit**, Frame Buffer can be allocated everywhere in the external Dynamic RAM area, such as SDRAM or DDR RAM.

### 11.4 Display

SPG290 can display arbitrary rectangle area within whole external Dynamic RAM area by setting the frame buffer start address register to the left-top pointer of the frame buffer which area is needed to be displayed ,

### 11.5 TVE Registers

SPG290 TV encoder unit provides two digital luminance filter, low pass filter, edge enhancement filter, and all pass filter to let user can depend on their content's characteristic to adjust the best TV signal output quality. For example, when displaying true color digital photos, enable low pass filter will let the image on TV screen looks more smoothly. On the contrary, the edge enhancement filter will be more suitable for the computer graphics. The all pass filter will bypass all digital filters, so user can place their own analog filter circuit on system PCB to adjust their idea TV output signal quality. In addition, the TV encoder unit can be disabled by programming to save the system power consumption whenever the TV composite output is not required, such as handheld applications.

#### 11.5.1 P\_TV\_Control

		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_TV_Control	8803_0000				TVEN							LPF_Mode		VGA	PAL	IntrIce	



		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_Fade_Control	8803_000C																

Fade\_offset

- FilterMode: Digital filter mode
- 00: Smoothness
  - 01: edge enhancement
  - 10: Normal
  - 11: Forbidden
- TVEn: TV encoder enable
- 0: disable
  - 1: enable
- Intrlace: Interlace mode select
- 0: non-interlace mode
  - 1: interlace mode
- PAL: NTSC/PAL system select
- 0: NTSC system
  - 1: PAL (B,D,G,H,I,N)system
- VGA: CIF/VGA mode select
- 0: CIF mode
  - 1: VGA mode

### 11.5.2 TV Saturation & Hue Adjust

Saturation & HUE level of TV encoder can be adjust by programming following two registers.  
Generally, they are initialized automatically when SPG290 boot. Be sure to leave them unchanged when running normal applications.

		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_TV_Saturation	8803_0004																
P_TV_Hue	8803_0008																

Saturation

Hue adjust



## 11.6 TV Encode API Services

SetupInterlace( )

SetupTVMode( )

SetupResolution( )

SetupLowPassFilter( )

### SetupInterlace( )

---

Prototype:               void SetupInterlace (UINT32 ptype)

Descriptions:           Setup TV interlace or non-interlace mode

Arguments:

ptype:	two type to select	
#define	Non-Interlace	0
#define	Interlace	1

Returns:                (None)

### SetupTVMode( )

---

Prototype:               void SetupTVMode (UINT32 tvtype)

Descriptions:           Setup TV NTSC/PAL mode

Arguments:

tvtype:	two type to select	
#define	NTSC	0
#define	PAL	2

Returns:                (None)



### SetupResolution( )

---

Prototype:                void SetupTVMode (UINT32 resolution)

Descriptions:            Setup TV resolution

Arguments:

    resolution: three type to select

        #define            QVGA\_Mode     0

        #define            VGA\_Mode      4

        #define            HVGA\_Mode     8

Returns:                (None)

### SetupLowPassFilter( )

---

Prototype:               void SetupLowPassFilter(UINT32 Level)

Descriptions:            Setup TV Low Pass Filter mode

Arguments:

    Level:               four type to select

        #define            Smooth\_Mode   0

        #define            AllPas\_Mode   16

        #define            Sharpness\_Mode 32

        #define            Edge\_Mode     48

Returns:                (None)

---

---

## 12 SOUND PROCESS UNIT - SPU

---

---

### 12.1 General Description

The Sound Processor built in the SPG290 is able to simulate all type of musical instruments by programming the tone ROM and controlling the envelope slope for each channel. Moreover, each channel can also be defined as a speech channel to provide percussion, animal sounds, gun, explosions and other sound effects in PCM format accompanied with the main music rhythm.

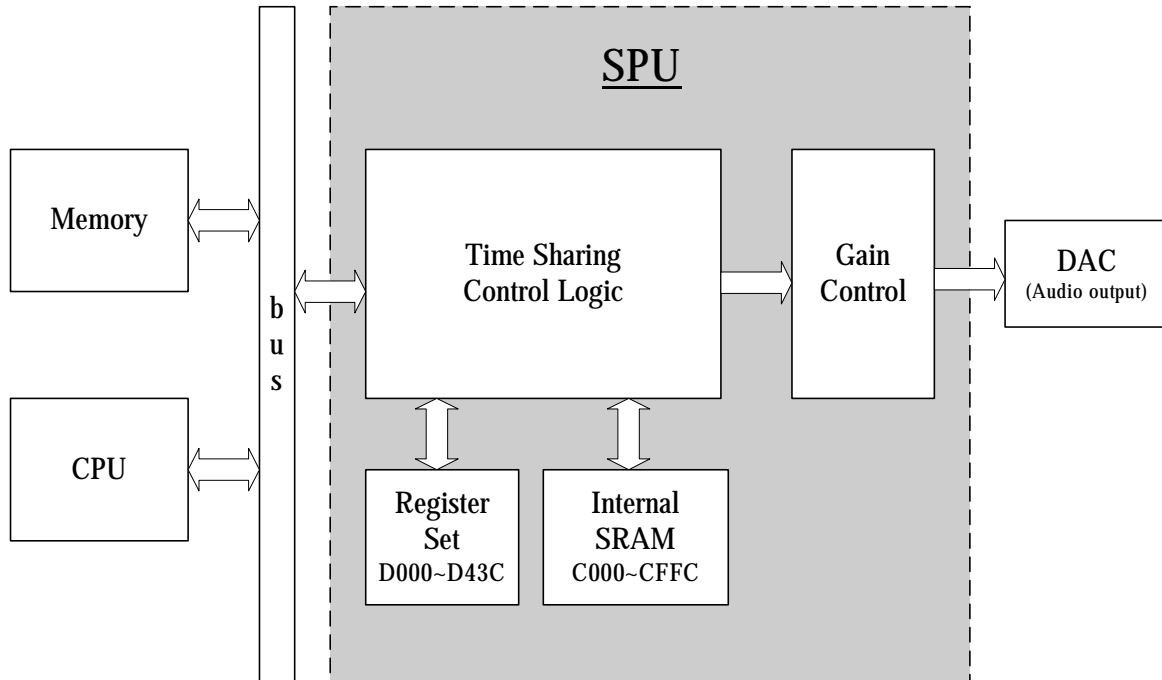
### 12.2 Feature

- I 8-bit /16-bit (software mode) stereo PCM
- I 4-bits ADPCM for each channel
- I 24 channels
- I Max. 4M x 16 addressing ability
- I 7 bits Main volume control
- I MIDI Format Gain control for each R/L channel of 24 channels
- I Max. 128 piece-wise slope with repeat for envelope control
- I 32-channel IRQ functions and Beat event IRQ and Envelope IRQ
- I Tone-Color & envelope address not limited by bank (64K). The Tone-Color & envelope can be controlled by hardware or manual (software) mode respectively.
- I Automatic tone-color interpolation function.
- I Max. 281.25kHz tone-color sample rate.
- I Hardware Release Tone Color function.
- I Individual Channel Release function.
- I Hardware Pitch Band function.
- I Automatically volume control function (Compressor).
- I 4-bands digital equalizer.





### 12.3 Block Diagram





## 12.4 Internal Memory Mapping

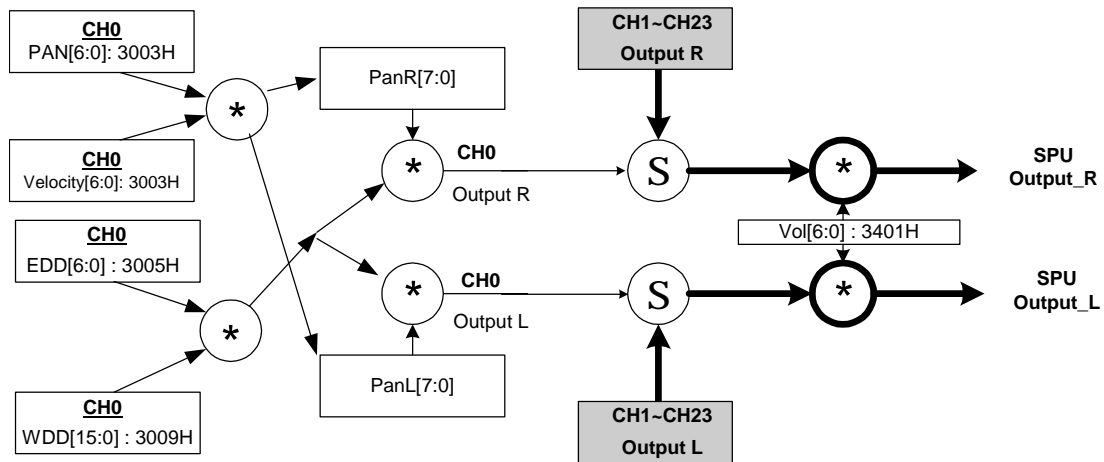
I Register Set: D000H ~ D068H(Channel 0~15), D400H ~ D460H(Channel 16~23)

I Internal SRAM: 24 channels

Channel	Phase Address Port(Hex)	Attribute Address Port(Hex)
0	C800~C83C	C000~C03C
1	C840~387C	C040~307C
2	C880~C8BC	C080~C0BC
3	C8C0~C8FC	C0C0~C0FC
4	C900~C93C	C100~C13C
5	C940~C97C	C140~C17C
6	C980~C9BC	C180~C1BC
7	C9C0~C9FC	C1C0~C1FC
8	CA00~CA3C	C200~C23C
9	CA40~CA7C	C240~C27C
10	CA80~CABC	C280~C2BC
11	CAC0~CAFC	C2C0~C2FC
12	CB00~CB3C	C300~C33C
13	CB40~CB7C	C340~C37C
14	CB80~CBBC	C380~C3BC
15	CBC0~CBFC	C3C0~C3FC
16	CC00~CC3C	C400~C43C
17	CC40~CC7C	C440~C47C
18	CC80~CCBC	C480~C4BC
19	CCC0~CCFC	C4C0~C4FC
20	CD00~CD3C	C500~C53C
21	CD40~CD7C	C540~C57C
22	CD80~CDBC	C580~C5BC
23	CDC0~CDFC	C5C0~C5FC



## 12.5 Gain Control



The SPU provides 24 channels. The above structure diagram only illustrates the Channel 0.

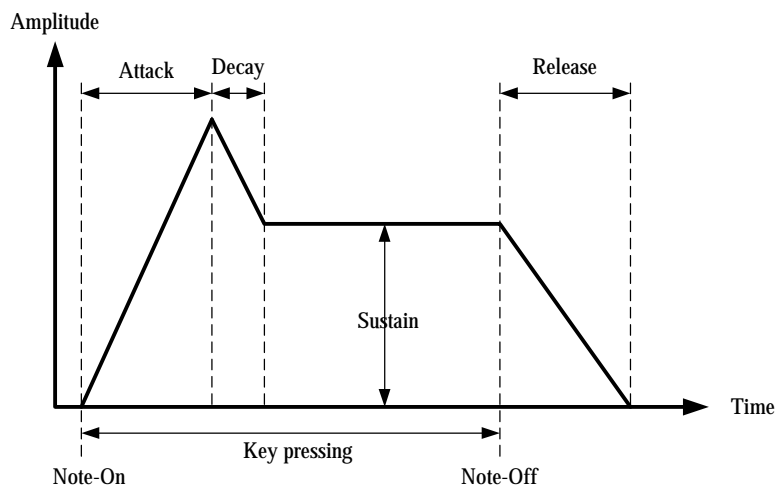
Similarities are applied for other channels. In the diagram, the multiplication result of tone-color and envelope is sent to each R/L channel to complete the panning effect. After all channels are added up, it outputs through the main volume control.

## 12.6 FUNDAMENTALS

Sound is composed of three essential elements: **pitch**, **tone-color**, and **envelope/ADSR**. The tone color can be classed to musical sound and noise in digital music.

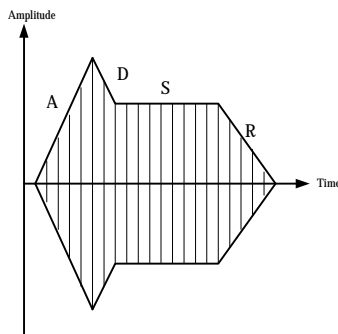
**Musical Sound:** With characteristics of cycles, vibration in certain pattern, e.g. piano, xylophone, violin, ...etc.

**Noise:** With the characteristic of vibration in random, e.g. animal sound.



### Musical sound

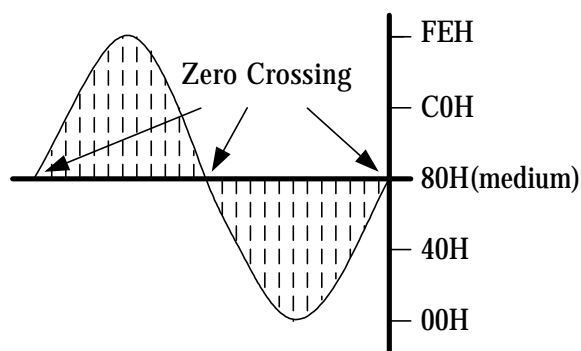
Since musical sound is a cycled wave, it is possible to be synthesized by repeating, which must be supported by hardware design. The instrument can be expressed by envelope and tone-color. The envelope can be four parts- **Attack**, **Decay**, **Sustain**, and **Release** or called **ADSR**, see the waveform above. See the waveform below for the modulations result from tone-color and envelope.



An envelope contains a series of repeated tone-color wave. The ADSR illustrated here is only a mode among all sound processing modes. Not all instruments are produced by this mode.

- I 8 bits tone-color code is expressed by unsigned. The medium is "80H"; max is "FEH" and min is "00H".  
The "FFH" is defined as the end code of tone-color; as a result, data itself cannot be "FFH".

### Sin Wave of Tone-Color



*Note: FFH defined as the end code of tone-Color*

- I The envelope data is expressed by 7 bits unsigned. The max value is "7FH", and min. value is "00H".
- I There are two envelop modes supported, envelope auto mode and manual mode.
  - a. **Envelope Auto Mode:** The CPU writes required parameters to **register set** and **internal SRAM**. Then, the hardware loads tone-color and envelope data automatically based on the given parameters. The envelope slope of an instrument can be up to 128 types. In addition, it has repeat function, which can be used to synthesize envelope LFO (Low Frequency Oscillation) easily.
  - b. **Envelope Manual Mode:** The envelope data is controlled by software.

## 12.7 Control Register

Address	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Description
D000H	ChEn[15:0]																Channel Enable
D004H										Vol[6:0]						Main volume	
D008H	ChFIQEn[15:0]																Channel FIQ Enable
D00CH	ChFIQSts[15:0]																Channel FIQ Status
D010H						BeatBaseCnt[10:0]											Beat base counter
D014H	BIE	BIS	BeatCnt[13:0]														Beat counter
D018H	EnvClk[15:0], Ch 3~0																Envelope interval select
D01CH	EnvClk[31:16], Ch 7~4																Envelope interval select
D020H	EnvClk[47:32], Ch 11~8																Envelope interval select
D024H	EnvClk[63:48], Ch 15~12																Envelope interval select
D028H	EnvRampDown[15:0]																Envelope fast ramp down
D02CH	ChStopSts[15:0]																Stop channel status
D030H	ChZeroCrossEn[15:0]																Zero cross Enable
D034H	Saturate			SoftCh	CompEn	NoHigh	NoInt	EQEn	VolSel		FOF		Init			Control Flags	
D038H	Peak	Threshold							AttScale		RelScale		DisZC	Ratio		Compressor Control	
D03CH	ChSts[15:0]																Channel status
D040H	WaveInL[15:0]																Left channel mixer input
	FIFO_Write_Data[15:0]																Software channel data input
D044H	WaveInR[15:0]																Left channel mixer input
	SoftIrq	SoftIrqEn								Phase_Soft[18:16]			FIFO_Irq_Threshold[3:0]			Software channel FIFO IRQ threshold	
D048H	WaveOutL[15:0]																Left channel mixer input
D04CH	WaveOutR[15:0]																Left channel mixer input
D050H	ChRepeatEn[15:0]																Channel Repeat Enable control
D054H	ChEnvMode[15:0]																Channel Env Mode
D058H	ChToneRelease[15:0]																Channel Tone Release Control
D05CH	ChEnvIrqSts[15:0]																Channel Env Irq Status
D060H	ChPitchBandEn[15:0]																Channel Pitch Band Enable
D064H	Phase_Soft[15:0]																Software channel phase
D068H	AttackTime								ReleaseTime								Attack/Release Time Control

D06CH		CutOff1[6:0]		CutOff0[6:0]	Digital EQ Cut-Off frequency 0/1
D070H		CutOff2[6:0]		CutOff2[6:0]	Digital EQ Cut-Off frequency 2/3
D074H		Gain1[6:0]		Gain0[6:0]	Digital EQ Gain 0/1
D078H		Gain3[6:0]		Gain2[6:0]	Digital EQ Gain 2/3
D07CH				BankAddr[8:0]	Wave Table's Bank Address

Note: All control registers are active high.

Address	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Description	
D400H									ChEn[23:16]								Channel Enable	
D404H																	-	
D408H									ChFIQEn[23:16]								Channel FIQ Enable	
D40CH									ChFIQSts[23:16]								Channel FIQ Status	
D410H																	-	
D414H																	-	
D418H									EnvClk[79:64], Ch 19~16								Envelope interval select	
D41CH									EnvClk[95:80], Ch 23~20								Envelope interval select	
D420H																		
D424H																		
D428H									EnvRampDown[23:16]								Envelope fast ramp down	
D42CH									ChStopSts[23:16]								Stop channel status	
D430H									ChZeroCrossEn[23:16]								Zero cross Enable	
D434H																		
D43CH									ChSts[23:16]								Channel status	
D440H																	-	
D444H																	-	
D448H																	-	
D44CH																	-	
D450H									ChRepeatEn[23:16]								Channel Repeat Enable control	
D454H									ChEnvMode[23:16]								Channel Env Mode	
D458H									ChToneRelease[23:16]								Channel Tone Release Control	
D45CH									ChEnvIrqSts[23:16]								Channel Env Irq Status	
D460H									ChPitchBandEn[23:16]								Channel Pitch Band Enable	





#### D000H; ChEn [15:0]

#### D400H; ChEn [23:16], Channel Enable Control and Status

1 : Channel Enable

0 : Channel Disable

I In H/W mode:

Before enabling a channel(ChEn [x]=1) , it is a **must** to initialize the Port[30x0H \* 4] ~ Port[30xFH \* 4].

Please refer to **Programming Note** section for initialization.

I To enable a channel, check the corresponding **ChStopSts** (D02CH) is cleared (writing "1" to ChStopSts corresponding bit) first. Write 0 to ChEn of a channel will set the ChStopSts of the channel to 1.

#### D004H; VOL [6:0]

Main volume control controls sound volume for entire system. It is designed that when all 24-channel is turn on with its max WDD, EDD, Velocity and global volumn, the output will not over flow.

#### D008H; ChFIQEn [15:0];

#### D408H; ChFIQEn [23:16]; Channel FIQ Enable

0 : Channel FIQ disable

1 : Channel FIQ enable.

Control FIQ enable. Refer to register D00CH.

#### D00CH; ChFIQSts [15:0]

#### D40CH; ChFIQSts [23:16]

Channel FIQ Status. ChFIQSts[x] occurs at the frequency of the sample rate of channel x, which depends on the channel phase.

Write 1 : Clear FIQ

Write 0 : No operation

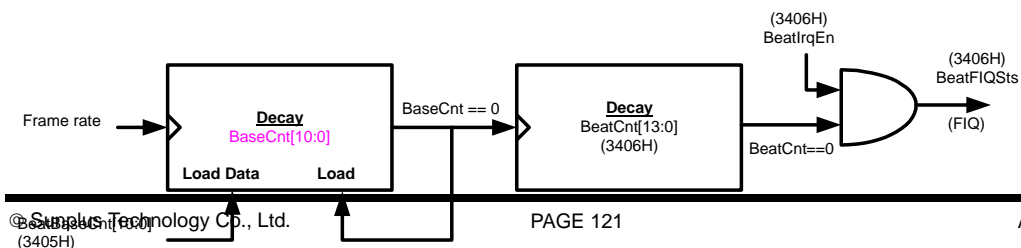
Read 1 : Channel FIQ active

Read 0 : Channel FIQ inactive

I FIQ occurs when the corresponding ChFIQEn[x] and ChFIQSts[x] be activated.

#### D010H; BeatBaseCnt [10:0]; Beat Base Counter (R/W)

b10	b9	b8	B7	b6	B5	b4	B3	b2	b1	b0
D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0





Beat count will be subtracted one every time when beat trigger count reach zero. Beat trigger count will be subtracted one every 4 frame.

#### D014H; Beat Counter

B15	b14	B13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BeatIRQEn	BeatIRQSts	BeatCnt[13:0]													

BeatIRQEn      **b15** (R/W): Beat IRQ Enable

0      Disable; Writing "0" to Beat(3105H) register clears the Beat FIQ.

1      Enable; When Beat event (Beat decay to 0) is enabled, FIQ activates.

BeatIRQSts      **b14** (R); Current Beat IRQ (IRQ 4) event status.

0      Beat IRQ (IRQ 4) no request

1      Beat IRQ (IRQ 4) request

BeatIRQ\_Period = ( **BeatBaseCnt** \* **BeatCnt** ) \* 14.208us.

**D018H, D01CH, D020H, D024H; EnvClk0[63:0];**  
**D418H, D41CH; EnvClk0[95:64]; Envelope Interval selection**

Ch0 : EnvClk[3:0]

Ch1 : EnvClk[7:4]

.....

.....

Ch23: EnvClk[95:94]

**I    Default:** ChxEnvClk0[1:0] = 0H

**I    ChxEnvClk0[1:0] where frame = 1/frame-rate**

0000 : EnvClk Count once / 4 \* 4 frame

0001 : EnvClk Count once / 8 \* 4 frame

0010 : EnvClk Count once / 16 \* 4 frame

0011 : EnvClk Count once / 32 \* 4 frame

0100 : EnvClk Count once / 64 \* 4 frame

0101 : EnvClk Count once / 128 \* 4 frame

0110 : EnvClk Count once / 256 \* 4 frame

0111 : EnvClk Count once / 512 \* 4 frame

1000 : EnvClk Count once / 1024 \* 4 frame

1001 : EnvClk Count once / 2048 \* 4 frame

1010 : EnvClk Count once / 4096 \* 4 frame

1011 : EnvClk Count once / 8192 \* 4 frame

1100 : EnvClk Count once / 8192 \* 4 frame

1101 : EnvClk Count once /  $8192 * 4$  frame

1110 : EnvClk Count once /  $8192 * 4$  frame

1111 : EnvClk Count once /  $8192 * 4$  frame

**D028H; EnvRampDown[15:0];**

**D428H; EnvRampDown[23:16]; For ChSts[x] control, see \$D03CH.**

When **EnvRampDown** is set to 1, **EnvRampDownClk** will be read from Phase SRAM and Ramp down offset will be read from attribute SRAM then the envelop will ramp down to 0 in 0~3sec depends on the **EnvRampDownClk** and **RampDownOffset**.

**C80CH, CC0CH; EnvRampDownClk:**

000: Envelope Ramp down one step every  $13 * 4 * 4$  frame = 0.738 ms

001: Envelope Ramp down one step every  $13 * 16 * 4$  frame = 2.955 ms

010: Envelope Ramp down one step every  $13 * 64 * 4$  frame = 11.821 ms

011: Envelope Ramp down one step every  $13 * 256 * 4$  frame = 47.284 ms

100: Envelope Ramp down one step every  $13 * 1024 * 4$  frame = 189.137 ms

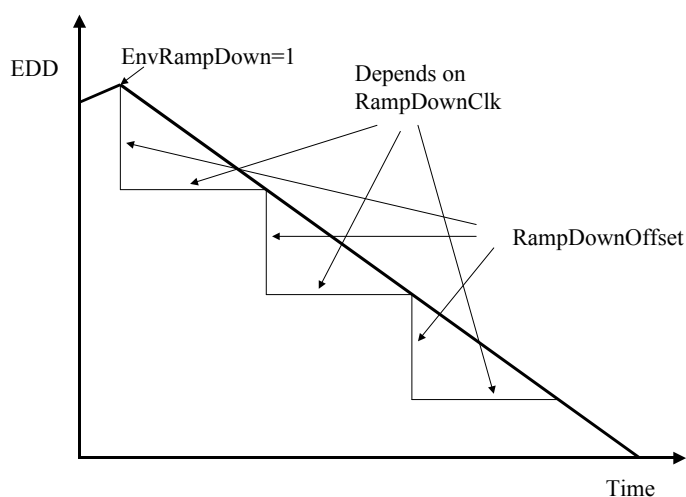
101: Envelope Ramp down one step every  $13 * 4096 * 4$  frame = 756.548 ms

110: Envelope Ramp down one step every  $13 * 8192 * 4$  frame = 1.513 s

111: Envelope Ramp down one step every  $13 * 8192 * 4$  frame = 1.513 s

**C028H, C428H; RampDownOffset**

The decreasing value of EDD at each ramp down step.



**D02CH; ChStopSts[15:0];**

**D42CH; ChStopSts[23:16]; channel stop status**



Write 1 : Clear Stop status

Write 0 : No operation

Read 1 : Channel is stopped

Read 0 : Channel is ready

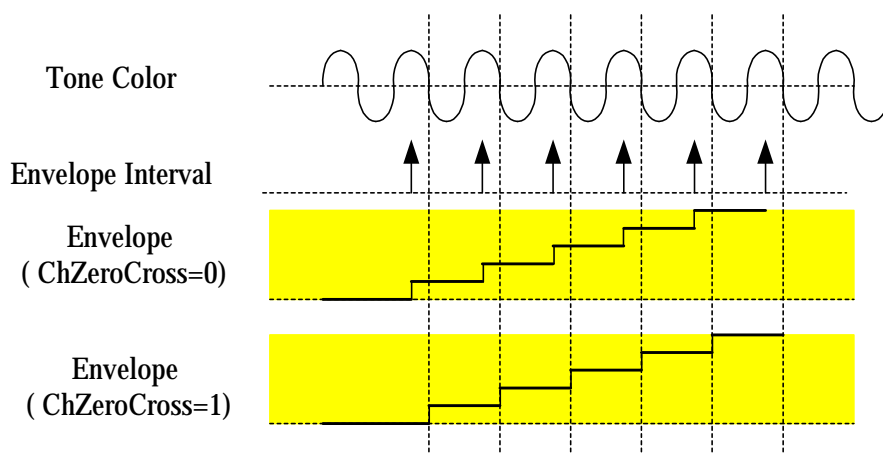
This stop bit is able to avoid channel to be ON accidentally. Before enabling (writing "1") channel [x], "1" must be written to the corresponding STOP ch[x]. Such process assures the channel is enabled successfully.

#### D030H; ChZeroCrossEn[15:0];

#### D430H; ChZeroCrossEn[23:16]; Channel zero crossing control.

0 : Envelope data changes according to the setup in Envelope Interval selection(D018 H – D024H, D418H – D41CH).

1: Envelope data changes only when the tone-color is at the zero-crossing point (default).



#### D034H Init

Sound processor channels accumulator initial control. To make sure that the system would be operated correctly, this initial command is required in the initialization of the sound processor. It's better to initialize the accumulator every time when the processor is enabled or disabled.

Write 0 : no operation

Write 1 : initial accumulator

#### D034H; FOF; FIFO over flow flag.

This flag is used to indicate the SPU is unable to deal with such high sample rate and cause data lost.

This bit is read only and write 1 can clear this bit. The max service rate of SPG290 is about **1500 kHz**, so

each channel can work without data lost at sample rate about  $2500/32\text{ch} = 78\text{ KHz}$ . Programmer can determine which channel needs more resource and which need few, if the total bandwidth needed is not excess 2500 kHz, data lost will not happen.

#### **D034H; VoISel; High volume select**

When VoISel is zero, the volume of a single channel will be 1/32 of max volume in 24-ch mode and 1/16 in 16-ch mode, 1/8 in 8-ch mode. When VoISel is set to a non-zero value, the volume of a single channel become larger according to the following table

VoISel Setting	Maximun Proportion of a Single Channel in Overall Output		
	24-ch Mode	16-ch Mode	8/6-ch Mode
00	1/32	1/16	1/8
01	1/8	1/4	1/2
10	1/2	1	2
11	1	2	4

#### **D034H; EQEn; Digital Equalizer Enable bit**

When LPEn is zero, the internal digital equalizer will be disabled, the output will be the original wave data.

When LPEn is one, the internal digital equalizer will be enabled.

#### **D034H; NoInt; Interpolation on/off control bit**

When NoInt is zero, the internal interpolation logic will used to smooth the output data when the phase is smaller than 0x10000, which cause the total bandwidth lower. If the FOF bit is set when program is running, set NoInt to one might help to improve the overall performance.

#### **D034H; NoHigh; High Quality Interpolation on/off control bit**

When NoHigh is zero, the internal interpolation logic will used to compensate the error induced by the phase-jitter effect, this will dramatically increase the sound quality of output wave data. Write 1 to this bit will disable this feature.

#### **D034H; SoftCh; Software channel enable bit**

When SoftCh is zero, the software channel will not have FIFO, i.e. the data written into D040H, D044H will be mixed with SPU's output directly. When SoftCh is one, an embedded 16-depth FIFO will be used for stored the wave data. Additionally, the setting in D044H and D064H will be used to control the output frequency of this software channel. The software channel have the capability to fired FIQ when data stored in FIFO in lower than threshold level set in D044H.

#### **D034H; Mixer\_Sel[1:0]; Software channel Left/Right mixer selection control**

These bits will only effected when SoftCh is set to 1.



Mixer\_Sel[1] :

0 : Software channel keeps silence output at Left channel.

1 : Software channel output the data written in FIFO at Left channel.

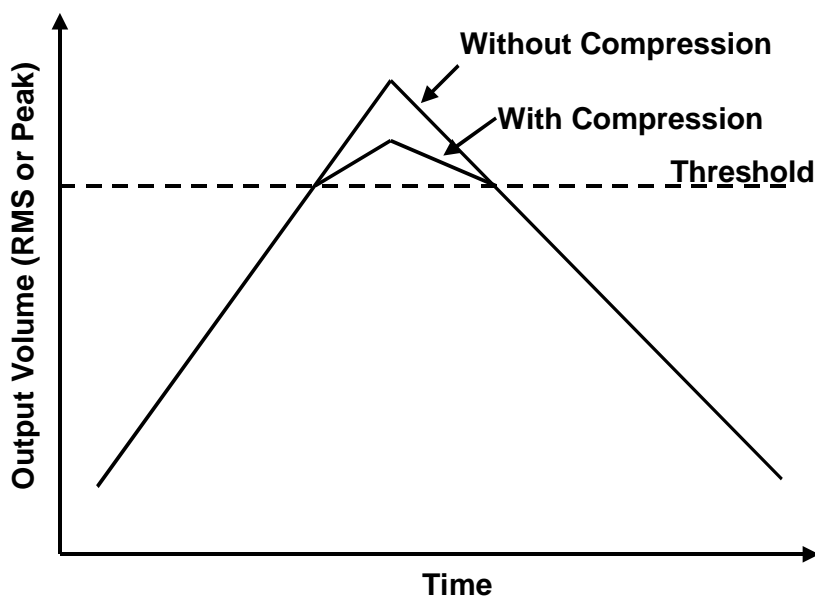
Mixer\_Sel[0] :

0 : Software channel keeps silence output at Right channel.

1 : Software channel output the data written in FIFO at Right channel.

#### D034H; CompEn; Compressor on/off control bit

When CompEn is one, the internal compressor will be activated and used the parameter stored in D038H and D064H to dynamically control the output volume. When the VolSel is set to a non-zero value, there is a possibility the output wave will saturate, the compressor can increase the over-all volume without saturate in output.



#### D034H; Saturate; Output signal saturation flag

This bit is used to indicate if the output signal is saturate. The saturate means the output signal is exceed the maximum range and the output will be clipped. When this bit is set to 1, it means a saturation condition is happened and it will not be cleared until the programmer write '1' to this bit. The saturate will happen only when VolSel is set to a non-zero value. To avoid this, use the compressor properly to increase the volume without affect the sound quality.

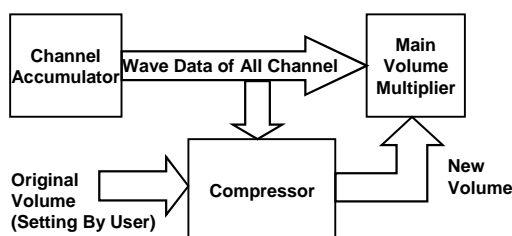
#### D038H; Peak; Peak/RMS Mode Selection of Compressor

When Peak is one, the compressor will detect the peak value of output wave data, when the peak value is larger than threshold value setting in Threshold register, the compressor will start to compress the output.

When Peak is zero, the compressor will detect the RMS value of output wave data. User can try both mode to determine which is better for the application.

#### D038H; Threshold; Threshold Level of Compressor

The compressor detects the output level “before” the main volume’s multiplier, and it will control the main volume dynamically. Look the following diagram for detail.



The threshold has maximum value 7FH and minimum value 01H, 00H is not allowed. Suppose the maximum value of wave data of all channels is 1, when peak or RMS of wave data of all channels is large than (threshold/80H), the compressor will start to active.

#### D038H; AttScale; Attack Time Scale Control Register

##### D064H; AttackTime[7:0]; Attack Time Register

The definition of attack time is how fast the compressor response for the wave data’s peak or RMS over the threshold. The fast attack time will result in fast response to the wave data, which means the compress will start in a short time. This is good for prevent the saturate but the transient part in the wave may be lost. So fast attack time is good for melody but bad for drum. The slow attack time will result in slow response to the wave data, which means the compress will start after a while. This is good for preserve the transient part in the wave but bad to prevent the saturation. Use can try both setting and find the best setting for the application.

The real attack time is 14-bits wide according to the following table

:

AttScale	Real Attack Time
00b	AttackTime[7:0] * 1
01b	AttackTime[7:0] * 4
10b	AttackTime[7:0] * 16
11b	AttackTime[7:0] * 64

Real attack time is 1 means the attack time is around 1.5 ms. Real attack time is 100 means the attack time is around 150 ms and so on.



**D038H; RelScale; Release Time Scale Control Register**

**D064H; ReleaseTime[7:0]; Release Time Register**

The definition of release time is how fast the compressor response for the wave data's peak or RMS lower than the threshold. The fast release time will result in the pumping of breathing effect happen, but the long release time will result in the over-all volume decrease. Use can try both setting and find the best setting for the application.

The real release time is 14-bits wide according to the following table

:

RelScale	Real Release Time
00b	$\text{ReleaseTime}[7:0] * 1$
01b	$\text{ReleaseTime}[7:0] * 4$
10b	$\text{ReleaseTime}[7:0] * 16$
11b	$\text{ReleaseTime}[7:0] * 64$

Real release time is 1 means the release time is around 1.5 ms. Real release time is 100 means the release time is around 150 ms and so on.

**D038H; DisZC; Disable Zero Cross Function of Compressor**

As described before, the compressor will adjust the main volume automatically. This bit is used to control if the volume change wait to the zero cross or not.

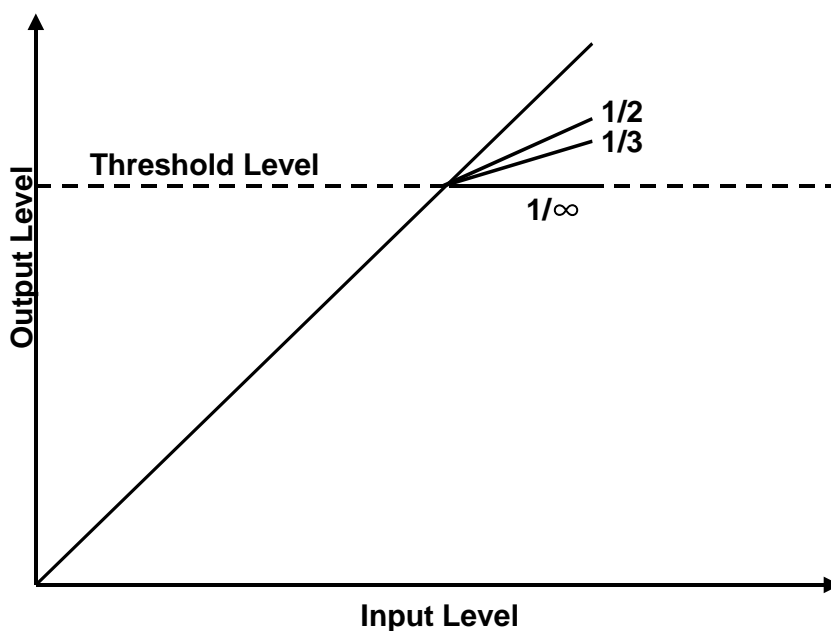
0 : Volume change will wait to the zero cross happen. We suggest use this configuration for better result.

1 : Volume change will not wait to the zero cross happen. Glitch sound might happen when the volume is changed.

**D038H; Ratio; Compress Ratio Setting**

The ratio is used to control how much does it compress when the level of output wave is high than the threshold level. Look the following figure for detail.





The following table shows the relationship between the ratio setting and the real compress ratio.

Ratio Setting	Compress Ratio
0	$1/2$
1	$1/3$
2	$1/4$
3	$1/5$
4	$1/6$
5	$1/7$
6	$1/8$
7	$1/\infty$ (Limiter)

Since the compress ratio is highly depended on the setting of VolSel, the following table can be used for selecting the suitable ratio.

**Suggested Threshold Level and Ratio Setting under different kind of VolSel Setting.**

VolSel	Minimum Channel Number to Cause Saturation			Effect	Suggested Threshold Level	Suggested Ratio Setting
	24-ch	16-ch	8/6ch			
00 (x1)	Not possible			-	Turn-off Compressor	Turn-off Compressor
01 (x4)	8	4	2	Compress	10H ( $1/8$ )	0 ( $1/2$ )
				Limit	20H ( $1/4$ )	7 ( $1/\infty$ )
10 (x16)	2	1	0.5	Compress	04H ( $1/32$ )	0 ( $1/2$ )
				Limit	08H ( $1/16$ )	7 ( $1/\infty$ )
11 (x32)	1	0.5	0.25	Compress	02H ( $1/64$ )	0 ( $1/2$ )
				Limit	04H ( $1/32$ )	7 ( $1/\infty$ )

User who wish to use compress ratio other than  $1/2$  and  $1/\infty$  can use the threshold level in between these

two setting to get the best result. The larger compress ratio can result in the overall volume reduce but with less saturation possibility.

**D03CH; ChSts [15:0];****D43CH; ChSts [31:16]; Channel status**

Read 0 : channel is valid

Read 1 : channel is busy

**D040H; WaveInL [15:0];****D044H; WaveInR [15:0]; An additional software channel.**

The data in these two port will be mixed with the SPU output.

**D048H; WaveOutL[15:0];****D04CH; WaveOutR[15:0]; The 16-bits output of SPU.**

The data in these two register is the final result of 24-channel + software channel when SoftCh is set to zero.

**D040H; FIFO\_Write\_Data [15:0]; Software Channel data out**

The data that would be output in software channel is written into this port sequentially into a 16-depth FIFO. And the data is read periodically according to the phase defined in **0xD064H** and **D044H**.

**D044H; FIFO\_Irq\_Threshold[3:0]; Software channel FIFO IRQ threshold**

When the number of the rest data is less than the FIFO\_Irq\_Threshold, the SoftIrqSts in D044H would be set.

**D044H; SoftIrqSts; Software channel FIQ Status**

This bit will be set when the number of the rest data is less than the FIFO\_Irq\_Threshold. The SPU will assert FIQ to CPU only when SoftIrqEn is set to 1.

**D044H; SoftIrqEn; Software channel FIQ enable bit**

0: Soft channel will not assert FIQ to CPU when the number of the rest data is less than the FIFO\_Irq\_Threshold.

1: Soft channel will not assert FIQ to CPU when the number of the rest data is less than the FIFO\_Irq\_Threshold.

**D044H; Phase\_Soft[18:16];**

**D064H; Phase\_Soft[15:0]; Software channel phase.**

This register is used to control the output period of software channel. The relationship between Phase\_Soft and sample rate is the same as that in hardware channel.

**D050H; ChRepeatEn [15:0];**

**D450H; ChRepeatEn [31:16]; Channel repeat enable.**

This register is used to control the repeat feature of envelope data.

0: Envelope repeat feature is disabled. RpEn and RpCnt is useless.

1: Envelope repeat feature is enabled(default). RpEn and RpCnt is usefull.

**D054H; ChEnvMode [15:0];**

**D454H; ChEnvMode [31:16]; Channel envelope mode.**

This register is used to control the envelope mode of each channel. Programmer can switch a channel from auto mode to manual mode any time. If programmer want to switch a channel from manual mode to auto mode, the attribute memory of the channel need to be programmed first.

0: Envelope of this channel is in manual mode.

1: Envelope of this channel is in auto mode.

**D058H; ChToneRelease [15:0];**

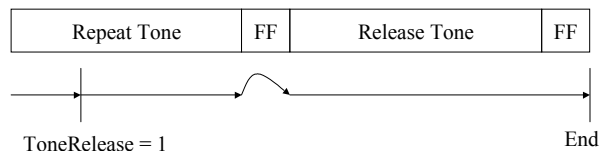
**D458H; ChToneRelease [31:16]; Channel tone release control register.**

This register is used to control the channel tone release feature. After write 1 to this bit, the channel will complete the current tone color and then begin to play the release tone then complete.

0: Do nothing.

1: Tone Release

This bit will be cleared automatically after channel stop.



**D05CH; ChEnvIrqSts [15:0]****D45CH; ChEnvIrqSts [31:16]; Channel envelope IRQ status.**

0: No Envelope IRQ.

1: Envelope IRQ is set.

Write 1 clear this bit. Envelope IRQ will be set if **IrqEn** is SRAM address 30X7H(31X7H) \* 4 is 1 and

**Eaoffset** match **IrqFireAddress** in 30X7H(31X7H) \* 4.

**D060H; ChPitchBandEn [15:0]****D460H; ChPitchBandEn [23:16]; Channel pitch band enable**

This register is used to control the pitch band feature of each channel.

0: Pitch band feature is disabled. TargetPhase[18:0], PhaseOffset[11:0], PhaseTimeStep[2:0], PhaseSign is discarded.

1: Pitch band feature is enabled. TargetPhase[18:0], PhaseOffset[11:0], PhaseTimeStep[2:0], PhaseSign will be used to increase/decrease phase. Please refer to **Internal SRAM** for detail.

**D06CH; CutOff0[6:0], CutOff1[6:0]****D070H; CutOff2[6:0], CutOff3[6:0]; Cut-off frequency setting of internal digital equalizer.**

These registers are used to control the cut-off frequency of internal digital equalizer. The internal digital equalizer have 4-bands and cut-off frequency of each band can be adjusted individually.

For CutOff0/CutOff1/CutOff2: 0 = 0Hz, 7F = 17.578125Hz

For CutOff3: 0 = 0Hz, 7F = 70.3125Hz

**D074H; Gain0[6:0], Gain1[6:0]****D078H; Gain2[6:0], Gain3[6:0]; Gain setting of internal digital equalizer.**

These registers are used to control the gain of internal digital equalizer. The internal digital equalizer have 4-bands and gain of each band can be adjusted individually.

00H: -12dB

40H: 0dB

7FH: 12dB

## 12.8 Internal SRAM

Internal Attribute SRAM Channel 0~15 Format:

Address	Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
C000H + x*64	Wave Address	Waddr[15:0]																
C004H + x*64	Mode	ADPCM	16M	ToneMode	LoopAddr[21:16]						Waddr[21:16]							
C008H + x*64	Loop Address	LoopAddr[15:0]																
C00CH + x*64	Pan		Pan[6:0]							ChVolumn[6:0]								
C010H + x*64	Envelope0	Repeat Period	EnvTarget[6:0]						EnvSig n		EnvInc[6:0]							
C014H + x*64	Envelope Data	EnvCnt[7:0]							EDD[6:0]									
C018H + x*64	Envelope1	RpCnt						Rpt	EnvLoad[7:0]									
C01CH + x*64	Envelope Address	IrqFireAddress[8:0]										IrqEn	Eaddr[21:16]					
C020H + x*64	Envelope Address	Eaddr[15:0]																
C024H + x*64	Wave Data 0	WDD0[15:0]																
C028H + x*64	Envelope Loop Control	RampDownoffset[6:0]						Eaoffset[8:0]										
C02CH + x*64	Wave Data	WDD[15:0]																
C030H + x*64	-																	
C034H + x*64	ADPCM Sel	ADPCM36	PointNumber															
C038H + x*64	-																	
C03CH + x*64	-																	

X = channel number

Internal Attribute SRAM Channel 16~23 Format:

Address	Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
C400H + x*64	Wave Address	Waddr[15:0]															
C404H + x*64	Mode	ADPCM	16M	ToneMode	LoopAddr[21:16]						Waddr[21:16]						
C408H + x*64	Loop Address	LoopAddr[15:0]															
C40CH + x*64	Pan		Pan[6:0]									ChVolumn[6:0]					
C410H + x*64	Envelope0	Repeat Period	EnvTarget[6:0]								EnvSig n	EnvInc[6:0]					
C414H + x*64	Envelope Data	EnvCnt[7:0]									EDD[6:0]						
C418H + x*64	Envelope1	RpCnt							Rpt	EnvLoad[7:0]							
C41CH + x*64	Envelope Address	IrqFireAddress[8:0]									IrqEn	Eaddr[21:16]					
C420H + x*64	Envelope Address	Eaddr[15:0]															
C424H + x*64	Wave Data 0	WDD0[15:0]															
C428H + x*64	Envelope Loop Control	RampDownoffset[6:0]								Eaoffset[8:0]							
C42CH + x*64	Wave Data	WDD[15:0]															
C430H + x*64	-																
C434H + x*64	ADPCM Sel	ADPCM36	PointNumber														
C438H + x*64	-																
C43CH + x*64	-																

X = channel number -16

Internal Phase SRAM Channel 0~15 Format:

Address	Name	D18	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
C800H + x*64	Phase	Phase[18:0]																		
C804H + x*64	Phase Accumulator	PhaseAcc[18:0]																		
C808H + x*64	Target Phase	Target Phase[18:0]																		
C80CH + x*64	Phase Control	RampDownClk			PhaseTimeStep				Sign	PhaseOffset										

X = channel number

Internal Phase SRAM Channel 16~23 Format:

Address	Name	D18	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
CC00H + x*64	Phase	Phase[18:0]																		
CC04H + x*64	Phase Accumulator	PhaseAcc[18:0]																		
CC08H + x*64	Target Phase	Target Phase[18:0]																		
CC0CH + x*64	Phase Control	RampDownClk			PhaseTimeStep				Sign	PhaseOffset										

X = channel number





This address is used for the tone color automatic repeat mode.

## C004H

b15	b14	b13 – b12	b11 – b6	b5 – b0
ADPCM	ToneColor16	ToneMode[1:0]	---	---

0: Wave data is treated as PCM data.

1: Wave data is treated as ADPCM data.

In ADPCM mode, user should set the ToneColor16 to one. Only Auto-End and Auto-Repeat mode is available in ADPCM mode. When the end code (0xFFFF) is reach, this bit will be cleared to zero automatically, i.e. back to PCM mode. This means only wave data in first region will be treated as ADPCM data, the following will be treated as PCM data. The PCM data can be 16-bits or 8-bits depends on the setting in ToneColor16. Two kinds of algorithm can be selected, please referred C034H for detail.

0 : 8-bits tone color data mode

1 : 16-bits tone color data mode

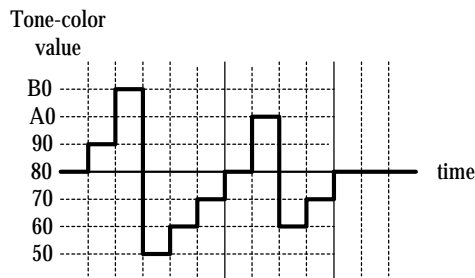
In 8-bits tone color data mode, wave data is accessed byte by byte. In 16-bits tone color data mode, data is accessed by word.

00 : S/W mode. Tone-Color S/W mode. In S/W mode, the tone color value is the value in WDD(C02CH, C42CH) that is written by CPU.

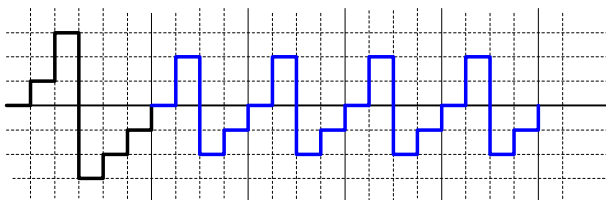
01 : H/W auto-end mode. Tone-Color H/W output with auto end. SPU auto load tone-color and stop playing when tone-color value is end-code (FFH for 8-bits tone color data mode and FFFFH for 16-bits tone color data mode).

10 : H/W auto-repeat mode. Tone-Color H/W output with auto repeat. SPU reload tone-color automatically when end-code is reached. The data between the loop address and the endcode would be repeat till the off of the channel.

Address	Data
WAddr + 0	9080H
WAddr + 1	50B0H
WAddr + 2	7060H
WAddr + 3	A080H
WAddr + 4	7060H
WAddr + 5	FFFFH



### Example : H/W auto-repeat mode



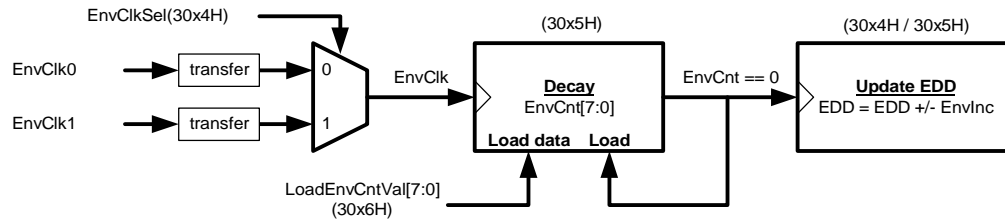
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	B2	b1	b0
---	EnvTargetValue [6:0]							Envsign	EnvInc [6:0]						

- 1: envelop increment direction is negative.



- I **EnvInc[6:0]**: envelop increment value for Envelop Tracking.

EDD = EDD +/- **EnvInc[6:0]** , depend on **EnvSign(0/1)** bit.



Note : "transfer" is EnvClkx mapping to EnvClk. Refer 3106H-3109H register description

#### C014H; Envelope Data

B15	b14	b13	b12	B11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
EnvCnt[7:0]								---	EDD [6:0]						

- I **EnvCnt[7:0]: Envelop Count**

The count is controlled by hardware in **Envelope automatic mode**. The clock source for envelop counter is from EnvClk. It records the period for counting envelop data (EDD) once.

- I **EDD[6:0]**

If envelope data reaches "0", the corresponding channel will be stopped automatically.

#### C018H; Envelope 1 for envelope tracking

b15	b14	b13	b12	b11	B10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RpCnt[6:0]							Repeat	EnvLoad[7:0]							

- I **Repeat** : Envelop repeat control

0 : Envelop normal mode

1 : Envelop repeat mode

In repeat mode, an extra word (**EnvelopLoopControl** in 30xAH) would be accessed from the external memory as shown in the following example. The repeat address(EAoffset) are stored in that word. The envelope would be repeated between the Eaoffset and repeat point RpCnt times.

- I **EnvLoad [7:0]: EnvLoad[7:0]** (C018H) is the value loaded into EnvCnt [7:0](C014H). After EnvCnt reaches "0", the EnvLoad is re-loaded automatically.

Initially, sound processor auto-loads EnvLoad[7:0] to internal register, **EnvCnt [7:0]**. The processor, according to the EnvClk0[1:0] definition or EnvClk1[1:0] in EnvClkSel (C010H), decays EnvCnt. When EnvCnt is decayed to "0", the processor will calculate the next EDD (envelop) value.  $EDD = EDD \pm EnvInc[6:0]$ , depend on **EnvSign(0/1)** bit. If EDD value reaches EnvTargetValue[6:0], the processor automatically update the data in envelope 0/1(Port C010H / C018H).

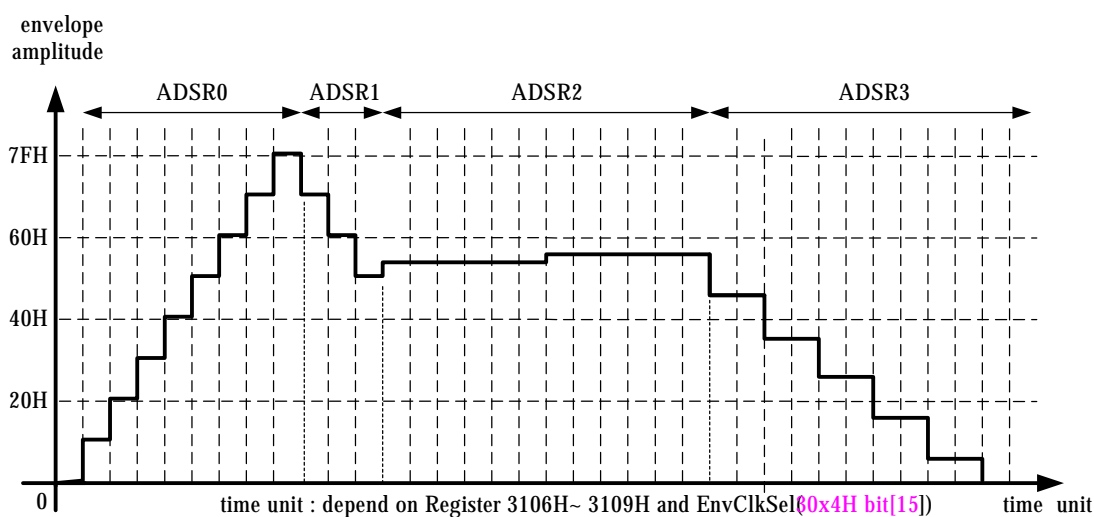
- I **RpCnt[6:0]: Envelop repeat counter**



The registers of C010H, C014H and C018H are used to control the envelope. In Envelope automatic mode(C004H), these registers are automatically reload/update by hardware from the external memory. In Envelope manual mode, CPU has to update the envelope data (EDD in C010H). To control the envelope in automatic mode, the envelope is controlled by Envelope0(C010H) and Envelope1(C018H) that is stored in external memory.

I Example for Envelop Format without Envelop Repeat

N <sub>th</sub> ADSR	Memory Offset[8:0] address	Envelop0/1 ( C010H/C018H)
0	0	Envelope 0
	1	Envelope 1
1	2	Envelope 0
	3	Envelope 1
2	4	Envelope 0
	5	Envelope 1
...	...	...
...	...	...



Suppose the waveform of envelop is shown as above. The start address of the envelope is as follows:

N <sub>th</sub> ADSR	Memory Offset[8:0] address	Envelop0/1 ( C010H/C018H)
0	0	7F10H
	1	0000H
1	2	5090H
	3	0000H
2	4	5804H
	5	0005H
3	6	0090H
	7	0001H

1. In ADSR0, it is increased from 0 to 7FH by 8 time units. Therefore, the following values are obtained:  
EnvTargetValue=7FH · EnvSign=0, EnvInc=(7FH-0)/8=10H, EnvCnt=0(EDD counts each time unit).



2. In ADSR1, it is decreased from 7FH to 50H by 3 time units. Therefore, the following values are obtained:  
EnvTargetValue =50H, EnvSign=1, EnvInc = (7FH-50H)/3=10H, EnvCnt =0(EDD counts each time unit).
3. In ADSR2, it is increased from 50H to 58H by 12 time units. Therefore, the following values are obtained:  
EnvTargetValue =58H, EnvSign=0, EnvInc=(58H-50H)/2=04H, EnvCnt =5 (EDD counts each 6 time units).
4. In ADSR3, it is decreased from 58H to 00H by 12 time units. Therefore, EnvTargetValue =00H, EnvSign=1,  
EnvInc=(58H-00H)/6=10H, EnvCnt =1(EDD counts each 2 time units).

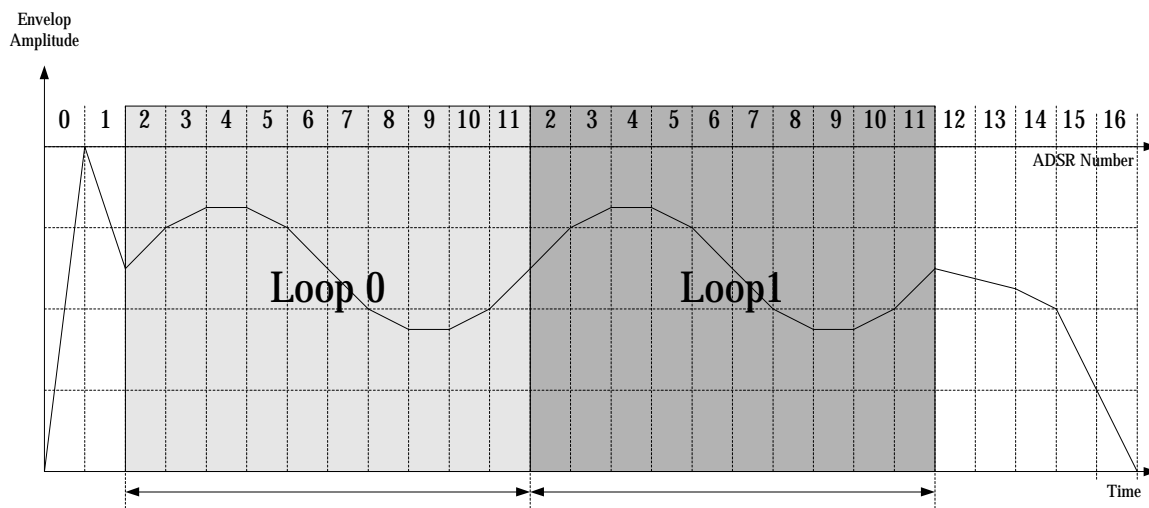
**Note:** For time unit definition, refer to **Envelope Interval selection**.

**I** Example for Envelop Format with Envelop Repeat

N <sub>th</sub> ADSR	Memory Offset[8:0] address	Envelop0/1 ( C010H/C018H)
0	0	Envelope 0
	1	Envelope 1
1	2	Envelope 0
	3	Envelope 1
2	4	Envelope 0
	5	Envelope 1
3	6	Envelope 0
	7	Envelope 1
4	8	Envelope 0
	9	Envelope 1
5	A	Envelope 0
	B	Envelope 1
6	C	Envelope 0
	D	Envelope 1
7	E	Envelope 0
	F	Envelope 1
8	10	Envelope 0
	11	Envelope 1
9	12	Envelope 0
	13	Envelope 1
10	14	Envelope 0
	15	Envelope 1
11	16	Envelope 0
	17	Envelope 1 (*1)
	18	Envelop Loop Control (*2)
12	19	Envelope 0
	1A	Envelope 1
...	...	...
	...	...

\*1: Repeat = 1.

\*2: In this example, EAoffset[8:0] = 04H, RpCnt[6:0]=01H.



#### C01CH, C020H (C41CH, C420H); Envelope Base Address

\$C01CH						\$C020H															
b5	b4	b3	b2	b1	b0	b15	B14	B13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
EAddr [21:0]																					

When envelope is set to auto mode, the H/W will use address in this register to get envelope data from external memory.

#### C01CH (C41CH); Envelope Irq Fire Address

b15	b14	b13	b12	b11	B10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IrqFireAddress[8:0]									IrqEn	EAddr[21:16]					

- I **IrqFireAddress[8:0]:** When Eaoffset(30xAH/31xAH) is match the IrqFireAddress and IrqEn is set to 1, the envelope IRQ(IRQ 4) will fired and EnvIrqSts[x](D05CH/D45CH) will be set.
- I **IrqEn:** Envelope IRQ enable bit.

#### C024H (C424H); WDD1[15:0]: Previous Tone-color data value (wave data)

B15	b14	B13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
WDD1[15:0]															

Programmer must write 0x8000 to this register when using the ADPCM36 mode, otherwise this register can be ignored.

#### C028H (C428H); Envelope Loop Control

b15	b14	b13	b12	b11	B10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RampDownOffset[6:0]								EAoffset[8:0]							

- I **RampDownOffset[6:0]:** Envelop ramp down offset, this value is used to decrease the EDD when EnvRampDown(D028H, D428H) of a channel is set. A read from external memory because of envelope

repeat will not affect this register.

- I EAoffset[8:0]:** . Envelop repeat address offset.

**C02CH (C42CH); WDD[15:0]:** Tone-color data value (wave data)

B15	b14	B13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
WDD[15:0]															

In tone color automatic modes (C004H), WDD is fetched automatically by hardware. In software mode, WDD is writte by CPU. When 8-bits tone color mode(C004H, C404H) is set, the wave data should be placed at **WDD[15:8]**. And in 16-bits tone color mode the tone color data is located in WDD[15:0].

**C034H; ADPCM Sel: ADPCM Mode Select**

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ADPCM36	PointNum[4:0]						Reserved								

- I ADPCM36:** When ADPCM bit in 30x1H is set to 1, use this bit to select between two kinds of algorithm.
- 0: Original algorithm in SPF32/SPF16/SPG200/SPG230, which has data rate around 32kbytes/sec.
  - 1: New algorithm with higher quality, which data rate around 36kbytes/sec.
- I PointNum[4:0]:** The point numbers of first frame in ADPCM36 mode, this field must be filled with the correct value before play a ADPCM36 tone colors. This field must be filled with 0 when the tonecolor is not in the ADPCM36 mode.
- 0: The first frame of ADPCM36 tone colors has 1 point.
  - 1: The first frame of ADPCM36 tone colors has 2 point.
  - 2: The first frame of ADPCM36 tone colors has 3 point.
  - 3: The first frame of ADPCM36 tone colors has 4 point.
- 31: The first frame of ADPCM36 tone colors has 32 point.



**C800H (CC00H); Phase [18:0]; control tone-color pitch (similar to sound pitch)**

B2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Phase[18:0]																		

Phase = sample-rate \*  $2^{19}$  / 281.25 kHz, for 54MHz Real Chip

Phase = sample-rate \*  $2^{19}$  / 140.625 kHz, for 27MHz FPGA

Although frame rate is not the same in 8- 16- 32-Ch mode, above formula is correct no matter what channel mode you use, Phase will be divided by 2 in 16-Ch mode and divided by 4 in 8-Ch mode automatically.

**C804H (CC04H); Phase Accumulator [18:0]; Tone-color pitch accumulator**

B2	b1	b0	b15	b14	b13	b12	b11	b10	b9	B8	b7	b6	b5	b4	b3	b2	b1	b0
PhaseAcc[18:0]																		

To have the lowest latency between the channel enable(ChEn[x], 3101H) and the play of the first tone color, it's suggested to fit the PhaseAcc[18:0] as the 7FFFFH.

**C808H (CC08H); TargetPhase [18:0]; Pitch Band Target Phase**

B2	b1	b0	b15	b14	b13	b12	b11	b10	b9	B8	B7	b6	b5	b4	b3	b2	b1	b0
TargetPhase[18:0]																		

The pitch band target phase, if PitchBandEn[x] is 1, the channel will use PhaseOffset[11:0] and PhaseSign and PhaseTimeStep to increase/decrease Phase to the TargetPhase.

**C80CH (CC0CH); Pitch Band Control and Ramp Down Control**

B15	b14	B13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PhaseTimeStep			Sign		PhaseOffset										

I PhaseOffset: This is the phase offset each time PhaseTimeStep is reached.

I PhaseSign: This is the increase decrease selection of phase.

0 : Increase phase.

1 : Decrease phase.

I PhaseTimeStep : This register is used to control the period of phase change.

000: Phase Change Every  $8 * 4$  frame = 0.114ms

001: Phase Change Every  $16 * 4$  frame = 0.227ms

010: Phase Change Every  $32 * 4$  frame = 0.455ms

011: Phase Change Every  $64 * 4$  frame = 0.909ms

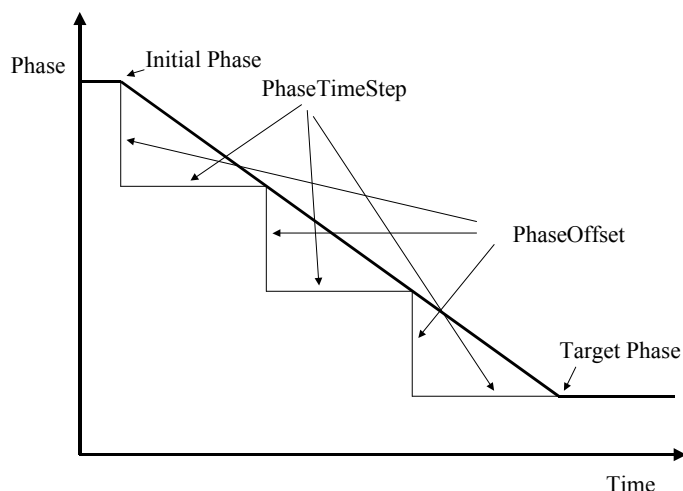
100: Phase Change Every  $128 * 4$  frame = 1.819ms

101: Phase Change Every  $256 * 4$  frame = 3.637ms

110: Phase Change Every  $512 * 4$  frame = 7.274ms

111: Phase Change Every  $1024 * 4$  frame = 14.549ms





#### Programming Notes

- I While initializing SPU SRAM, data must be written into Attribute SRAM and Phase SRAM. The Port[30x5H] and Port[30xAH] should be set to zero. For 8 bits tone color application, Port[30x9H] = 8000H (middle level), and Port[30xBH] = 8080H or the first tone-color. For 16 bits tone color application, Port[30x9H] = 8000H (middle level), and Port[30xBH] = 8000H or the first tone-color. Besides, "1" should be written into Port[310DH].b3 to set up the related SRAM initial value.

- I Note: When update data during beat event, make sure the corresponding bit is "0" in 3101H (ChSts).

**Otherwise, hardware control is unpredictable.**

There are three methods to disable ChSts[x] :

1. Disable ChEn[x] and the channel would be disable in the next tone-color zero-crossing.
2. S/W writes EnvRampDown[x]=1. H/W will ramp down in speed defined in EnvRampDownClk[2:0] and offset in EnvRampDownOffset[6:0] until envelope is equal to zero
3. When the envelop data EDD(30x5H) is 0.
4. When the tone color reaches the end point(FFH) in the H/W auto-end mode.

#### I Frame Rate

For Real Chip

32-Ch mode: Frame-Rate =  $54\text{MHz}/(32\text{Ch} \cdot 6\text{T}/\text{Ch}) = 281.25\text{kHz}$

16-Ch mode: Frame-Rate =  $54\text{MHz}/(16\text{Ch} \cdot 6\text{T}/\text{Ch}) = 562.5\text{kHz}$

8-Ch mode: Frame-Rate =  $54\text{MHz}/(8\text{Ch} \cdot 6\text{T}/\text{Ch}) = 1125\text{kHz}$

For 27MHz FPGA Chip

32-Ch mode: Frame-Rate =  $27\text{MHz}/(32\text{Ch} \cdot 6\text{T}/\text{Ch}) = 140.625\text{kHz}$

16-Ch mode: Frame-Rate =  $27\text{MHz}/(16\text{Ch} \cdot 6\text{T}/\text{Ch}) = 281.25\text{kHz}$

8-Ch mode: Frame-Rate =  $27\text{MHz}/(8\text{Ch} \cdot 6\text{T}/\text{Ch}) = 562.5\text{kHz}$

For convenient, the frame time is set to same the in all mode when using by envelop count or phase band count which is independent to the channel mode.

For Real Chip

A frame = 3.55 us

For 27 MHz FPGA Chip

A frame = 7.11 us

#### C80CH (CC0CH); Envelope Ramp Down Clock Selection Control

													b18	b17	b16
---													EnvRampDownClk		

This register is used to control the ramp down period when EnvRampDown[x] is set to 1.

000: Envelope Ramp down one step every  $13 * 4 * 4$  frame = 0.738 ms

001: Envelope Ramp down one step every  $13 * 16 * 4$  frame = 2.955 ms

010: Envelope Ramp down one step every  $13 * 64 * 4$  frame = 11.821 ms

011: Envelope Ramp down one step every  $13 * 256 * 4$  frame = 47.284 ms

100: Envelope Ramp down one step every  $13 * 1024 * 4$  frame = 189.137 ms

101: Envelope Ramp down one step every  $13 * 4096 * 4$  frame = 756.548 ms

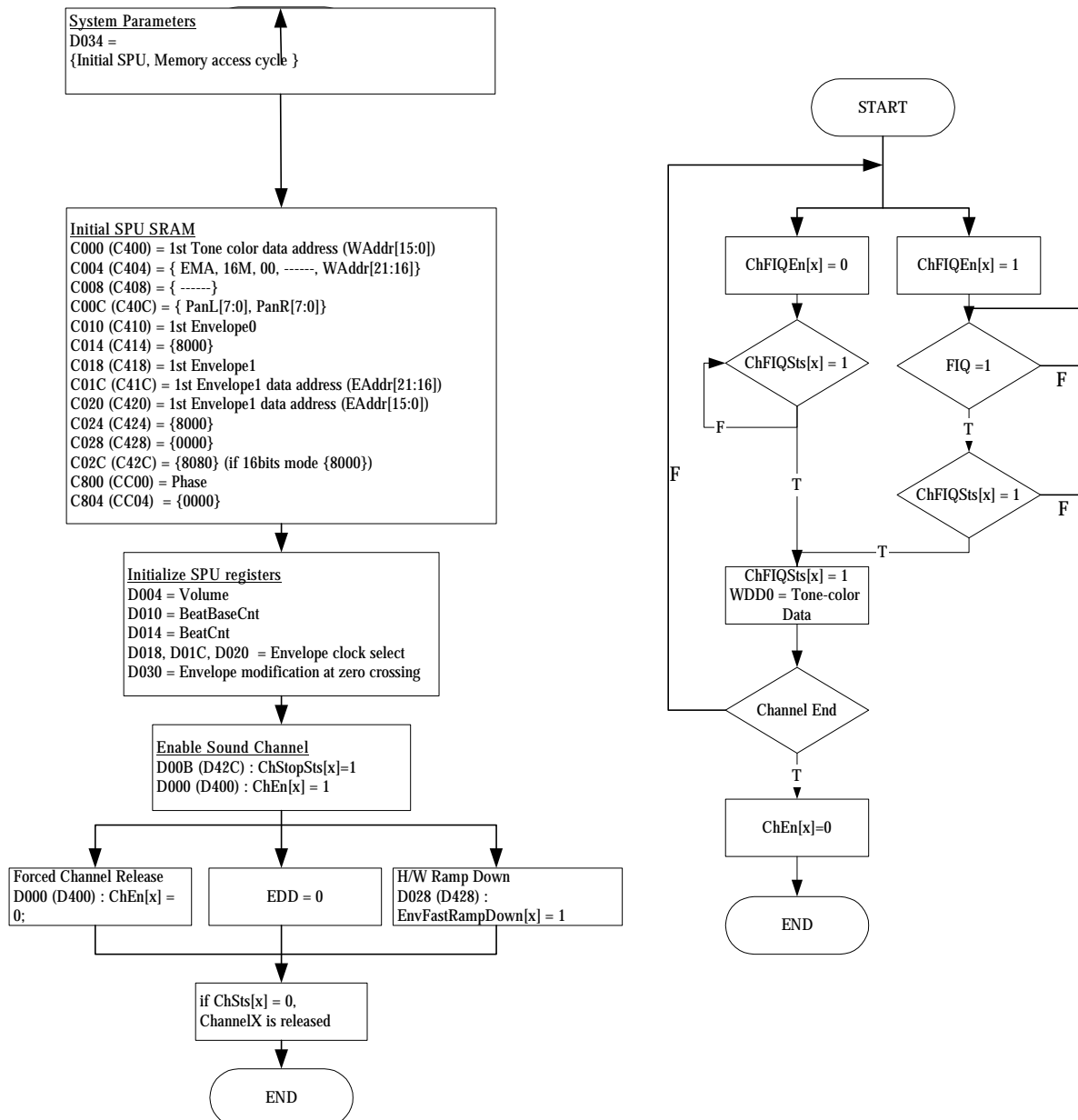
110: Envelope Ramp down one step every  $13 * 8192 * 4$  frame = 1.513 s

111: Envelope Ramp down one step every  $13 * 8192 * 4$  frame = 1.513 s



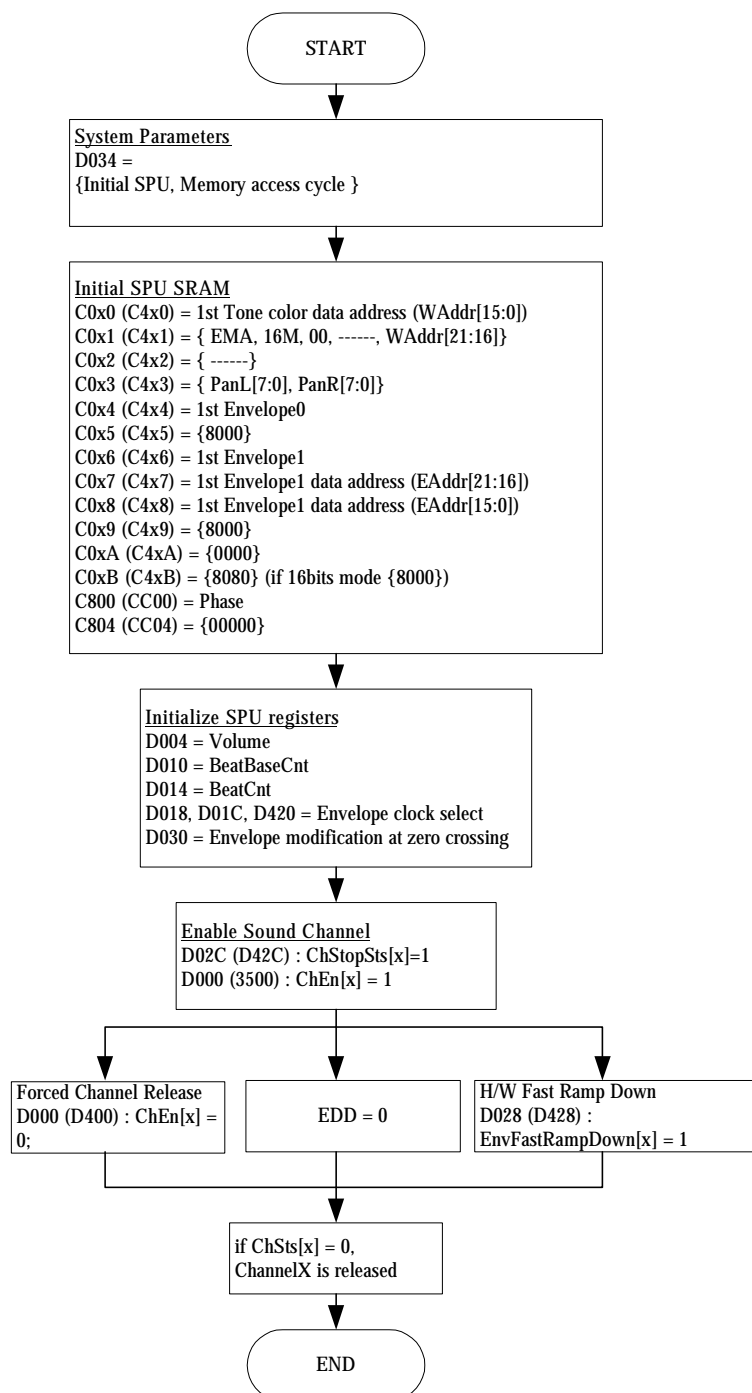
## 12.9 S/W Programming Flow Example

### I S/W mode:





I H/W mode





---

---

## 13 MP4 CODEC

---

---

## 14 CMOS SENSOR APPLICATION

### 14.1 Frame buffer

Frame buffer 一定是位於 external DRAM 裡，因為 embedded SRAM 容量不足以儲存整張 frame。CSI 內有三個 CSI frame buffer start address register 作為 triple buffer 之用，address 由 CPU 從 AHB slave 填入。CSI 會把 local address 加上 CSI frame buffer start address，至於 CSI 要讀寫哪一個 frame buffer 是由 BUFCTL module 控制。CSI 輸出的 HADDR[31:0] 是 DRAM 的絕對位址。

Frame buffer 內的資料是採 raster-scan 順序存放。一條 scan line 有幾個 pixel 亦記錄於 control register 中。CSI 每次從 AHB 寫出的 image data 一定是 32-bit word，一個 32-bit word 內含 YUYV 或是 RGB565，代表 2 個 pixel 的資料。

### 14.2 Buffer control

CSI 寫到 DRAM 的 frame 會由 PPU、TV、LCD、JPG 讀出，這些 module 的 frame rate 未必與 CSI 相同或是有倍數關係，所以 CSI 與這些 module 之間必須做 buffer control，以避免 data loss。所有的 buffer control 都交由 BUFCTL 這個 module 負責。對 CSI 而言，只要有以下的 output port 即可：

Name	I/O	Function description
CSI_FRAME_END	output	A pulse to indicate that CSI has completed the write transfer of a frame. Every frame must have a FRAME_END pulse, even the frame is disabled by the DISABLE_FRAME input.

CSI\_FRAME\_END 是 AHB master clock 驅動。

BUFCTL module 有以下的 I/O port：

Name	I/O	function description
CSI_FRAME_END	Input	A pulse to indicate that CSI has completed the write transfer of a frame. Every frame must have a FRAME_END pulse, even the frame is disabled by the DISABLE_FRAME input.
CSI_BUFFER_PTR[1:0]	Output	2'b00: write CSI frame buffer A 2'b01: write CSI frame buffer B 2'b10: write CSI frame buffer C
DISABLE_CSI_FRAME	Output	Disable the write transfer for a whole frame, CSI holds HTRANS at IDLE. This input changes after the CSI_FRAME_END pulse.

CSI\_BUFFER\_PTR 接到 MIU，DISABLE\_CSI\_FRAME 接到 CSI。

### 14.3 Bus interface

CSI 有兩個 AHB master 及一個 AHB slave。第一個 AHB master 目的是要讓 CSI access frame buffer（位於 external DRAM），第二個 AHBmaster 是要讓 CSI access motion detection 使用的 buffer（位於 external DRAM）。AHB slave 是要讓 CPU 讀寫 CSI 中的 control registers。爲了以後這個 IP 能給其他不同的計畫使用，AHB master 和 slave 都必須做成 fully compliant with AMBA 2.0。

對 SPG290 而言，兩個 CSI AHB master 只接到 MIU（memory interface unit）上專屬的 AHB slave，不需要 Arbiter，所以 HGRANT input 會固定接 1，而 HBUSREQ output 會 float。CSI AHB slave 不做 RETRY/SPLIT（沒必要），所以 HMASTER[3:0]，HMASTLOCK，HSPLITx[15:0]這些 port 都不必做。

CSI AHB master 每次 request 一定是 4-beat incrementing burst 或 single transfer：

Image frame buffer           => 4-beat incrementing burst，一次 burst access 16 bytes

Motion detection buffer     => single transfer，一次 access 4 bytes

也就是 HBURST 只有以下兩種

HBURST[2:0] = 3'b011（4-beat incrementing burst）

HBURST[2:0] = 3'b000（single transfer）

CSI AHB master 的 HSIZE[2:0] 一定是 3'b010（32 bits），CSI AHB slave 的 HSIZE[2:0]有 3'b000（8 bits），3'b001（16 bits）和 3'b010（32 bits）三種可能。

AHB master clock frequency 與 CSI main clock frequency 沒有關係，但一定高於或等於 27MHz。

AHB slave clock frequency 與 CSI main clock frequency 沒有關係，但一定高於或等於 27MHz。

### 14.4 Control registers

CSI 中的 control registers 係由 CPU 從 AHB slave 寫入/讀出。

CSI 必須有一個 disable reg，當 CPU 填 1 時，CSI 會等目前對 MIU 的 AHB transfer 完成後停止 AHB transfer。停止 AHB transfer 後 CSI 會以 status register 的一個 bit 讓 CPU 知道已停止對 MIU 的 AHB transfer。

### 14.5 Time Generator

In order to connect specific sensor, the parameters of TG are necessary to be setup. There are several parameters that are necessary to be setup: TG\_CR, TG\_LSTART, TG\_START, TG\_END.



#### 14.5.1 P\_TG\_CR (0x08000000): Time Generator Control

NAME	D7	D6	D5	D4	D3	D2	D1	D0
P_TG_CR	BSEN	YUVOUT	YUVIN	CLKINV	RGB565	VGAEN	MASEN	CSIEN
NAME	D15	D14	D13	D12	D11	D10	D9	D8
P_TG_CR	INTERLACE	FIELDSEL	YUV_TYPE	VRST_TYPE	VADD_TYPE	HRST_TYPE	FGET_TYPE	CCIR656
NAME	D23	D22	D21	D20	D19	D18	D17	D16
P_TG_CR					RESIZE		D_TYPE	

CSIEN: Sensor interface enable register

0: Disable

1: Enable

MASEN: SPG Master/Slave mode select

0: SPG Slave mode(if connected with OV7648/OV7649)

1: SPG master mode(if connected with S202/S201)

VGAEN: When SPG is slave mode (MASEN is 0)

0: QVGA(each line contains 320 pixels)

1: VGA(each line contains 640 pixels)

When SPG is master mode(MASEN is 1)

0: S202 is connected

1: S201 is connected

RGB565: The OV7648 has a RGB565 mode output, this register is used to control it is RGB888 or RGB565 mode.

0: RGB888 mode

1: RGB565 mode

This bit is valid only when MASEN is 0 and YUVIN is 0

CLKINV: Sensor Clock inverse select

0: Non-inverse input sensor clock

1: Inverse input sensor clock

YUVIN: YUV input mode

0: Input data is RGB

1: Input data is YUV

This bit is valid only when MASEN is 0(i.e. OV7648/OV7649 is connected)

YUVOUT: YUV output mode

0: Output data is RGBRGB

1: Output data is YUV422

BSEN: Blue Screen effect select

0: Disable Blue Screen Effect

1: Enable Blue Screen Effect

CCIR656: CCIR601/CCIR656 select





0: CCIR601 Interface

1: CCIR656 Interface

This bit is valid only when MASEN is 0.

FGET\_TYPE: Field get type

0: Referring to field at falling edge of VSYHC

1: Referring to field at rising edge of VSYHC.

It's set to 1 if connected with OV7648

HRST\_TYPE: Horizontal counter

0: Reset at the falling edge of HSYNC

1: Reset at the rising edge of HSYNC

VADD\_TYPE: Vertical counter

0: Add at the falling edge of HSYNC

1: Add at the rising edge of HSYNC

VRST\_TYPE: Vertical counter

0: Reset at the falling edge of VSYNC

1: Reset at the rising edge of VSYNC

YUV\_TYPE: Sensor data sequence

0: UYVY(BGRB)

1: YUYU(GBGR)

FIELDSEL: The polarity of input FODD

0: Reversing is not necessary

1: Reverse the FODD signal

INTERLACE: Non-Interlace/Interlace mode select

0: Non-Interlace mode(Field is don't care)

1: Interlace mode

D\_TYPE: The Parameter of data sequence control

RESIZE: Re-Size frame buffer control

0: No re-size is applied

1: VGA => QVGA re-size will be applied when VGA sensor is connected

#### 14.5.2 P\_TG\_LSTART(0x08000004):

NAME	D29-D20	D19-D10	D9-D0
P_TG_LSTART	TG_VL1START	TG_VL0START	TG_HLSTART

TG\_HLSTART: The parameter of horizontal TG.

To get the actual starting pixel data at a line from sensor.

TG\_VL0START: The parameter of horizontal TG.

To get the actual starting line from sensor in field 0 (interlace mode) or in

each frame (non-interlace mode)

TG\_VL1START: The parameter of horizontal TG.

To get the actual starting line from sensor in field 1 (interlace mode).

#### 14.5.3 P\_TG\_START(0x08000008):

NAME	D24-D16	D15-D10	D9-D0
P_TG_START	TG_VSTART		TG_HSTART

TG\_HSTART: The horizontal start of a sensor window.

If it's 0, the left side contains 0 black column. If it's 20, the left side contains 20 black columns where the black color is defined in TG\_BLACK.

TG\_VSTART: The vertical start of a sensor window.

If it's 0, the top side contains 0 black row. If it's 10, the top side contains 10 black rows where the black color is defined in TG\_BLACK.

#### 14.5.4 P\_TG\_END(0x0800000C):

NAME	D24-D16	D15-D10	D9-D0
P_TG_END	TG_VEND		TG_HEND

TG\_HEND: The horizontal end of a sensor window.

For example, in VGA mode. If it's 640, the right side contains 0 black column. If it's 620, the right side contains 20 black lines where the black color is defined in TG\_BLACK.

TG\_VEND: The vertical end of a sensor window.

If it's 0, the bottom side contains 0 black row. If it's 10, the bottom side contains 10 black rows where the black color is defined in TG\_BLACK.

#### 14.5.5 P\_TG\_BLACK(0x0800000C):

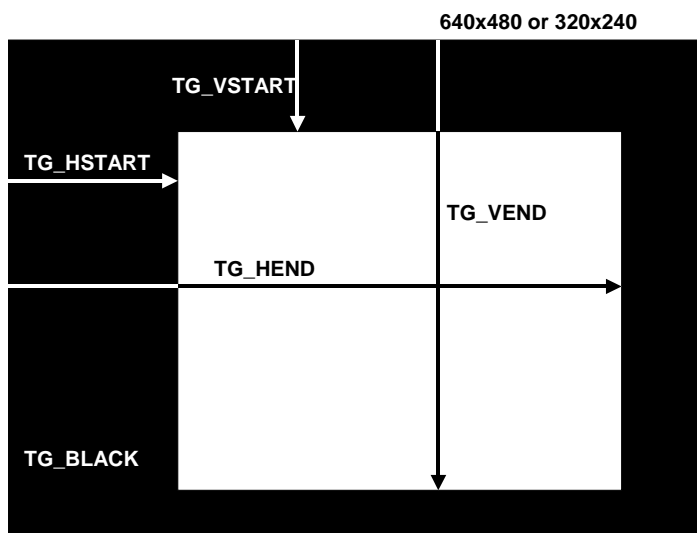
NAME	D23-D0
P_TG_BLACK	TG_BLACK

TG\_BLACK: The Black color definition.

When YUVOUT is 0, this is RGB888.

When YUVOUT is 1, this is YUV888

The following diagram shows the definition of these registers.



#### 14.5.6 P\_TG\_BSUPPER(0x08000014):

NAME	D23-D16	D15-D8	D7-D0
P_TG_BSUPPER	TG_BSUPPER_B	TG_BSUPPER_G	TG_BSUPPER_R

TG\_BSUPPER\_R: The upper boundary of Red Color for Blue screen detection.

TG\_BSUPPER\_G: The upper boundary of Green Color for Blue screen detection.

TG\_BSUPPER\_B: The upper boundary of Blue Color for Blue screen detection.

#### 14.5.7 P\_TG\_BSLOWER(0x08000018):

NAME	D23-D16	D15-D8	D7-D0
P_TG_BSLOWER	TG_BSLOWER_B	TG_BSLOWER_G	TG_BSLOWER_R

TG\_BSLOWER\_R: The lower boundary of Red Color for Blue screen detection.

TG\_BSLOWER\_G: The lower boundary of Green Color for Blue screen detection.

TG\_BSLOWER\_B: The lower boundary of Blue Color for Blue screen detection.

The blue screen region is defined in the following equation.

$(TG\_BSLOWER\_R < R/(R+G+B) < TG\_BSUPPER\_R)$  AND

$(TG\_BSLOWER\_G < G/(R+G+B) < TG\_BSUPPER\_G)$  AND

$(TG\_BSLOWER\_B < B/(R+G+B) < TG\_BSUPPER\_B)$  AND BSEN



#### 14.5.8 P\_TG\_TRANSP(0x0800001C):

NAME	D23-D0
P_TG_TRANSP	TG_TRANSP

**TG\_TRANSP:** The transparent color definition.

When YUVOUT is 0, this is RGB888.

When YUVOUT is 1, this is YUV888.

This register will be used as the output data of sensor when a pixel is defined as the blue screen region.

#### 14.5.9 P\_TG\_FBSADDR1(0x08000020): Frame Buffer 1's start address

NAME	D31-D0
P_TG_FBSADDR1	TG_FBSADDR1

#### 14.5.10 P\_TG\_FBSADDR2(0x08000024): Frame Buffer 2's start address

NAME	D31-D0
P_TG_FBSADDR2	TG_FBSADDR2

#### 14.5.11 P\_TG\_FBSADDR3(0x08000028): Frame Buffer 3's start address

NAME	D31-D0
P_TG_FBSADDR3	TG_FBSADDR3

**PS:** The frame buffer control is outside of sensor interface, the external hardware will determine which frame buffer is going to be written.

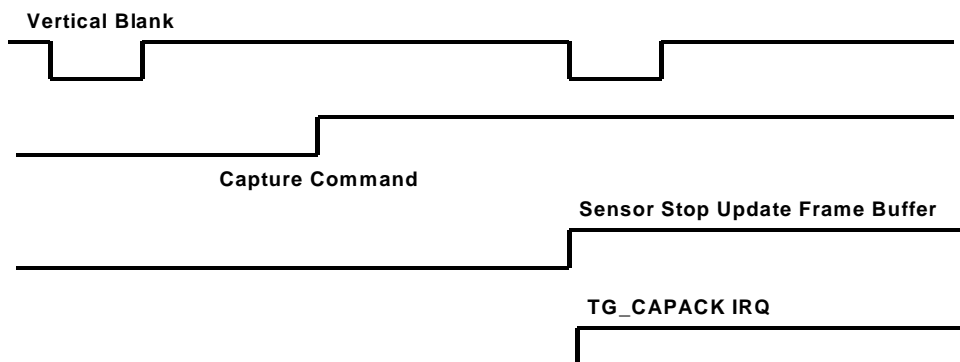
#### 14.5.12 P\_TG\_CAP(0x0800002C): Frame Buffer 3's start address

NAME	D0
P_TG_CAP	TG_CAP

TG\_CAP: Capture control bit.

- 0: the sensor will return to normal function.
- 1: the sensor will stop update frame buffer after the current frame is updated completely. An interrupt will be asserted after the sensor is stop.

The following figure shows the capture flow.



### 14.6 Motion Detect Control

#### 14.6.1 P\_MD\_CR(0x88000030): Motion Detect control register.

NAME	D15-D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_MD_CR	DIFFY_THRES	SAMP_TYPE	CC_WHITE	CC_BLACK	FRAME_TYPE	MD_TYPE				

MD\_TYPE: Motion detect type selection register.

- 00: Motion detect function disable.
- 01: Reserved
- 10: SPG220's Y difference detect function.
- 11: SPG220's Y difference detect function and color classification function.

FRAME\_TYPE: Control the sampling frequency of motion detect function

- 00: Sample in each frame.
- 01: Sample in every two frame.



10: Sample in every four frame.

11: Sample in every eight frame.

CC\_BLACK: Black color definition of color classification circuit.

CC\_WHITE: White color definition of color classification circuit.

SAMP\_TYPE: Motion Detect size select

0 : 16x16

1 : 8x8, when MD\_TYPE is 0x11 and VGA mode is selected, this mode is not allowed.

DIFFY\_THRES: The recognition function is:

$$\text{DIFF} = |\text{Y}_{\text{new}} - \text{Y}_{\text{old}}|/2 \geq \text{DIFFY\_THRES}$$

Notes: DIFF bit is stored in address defined MD\_SADDR, for each word in memory:

{Y3[7:1], Y2[7:1], Y1[7:1], Y0[7:1], DIFF3, DIFF2, DIFF1, DIFF0}.

Sample 2 is at the right side of sample 1.

#### 14.6.2 P\_MD\_SADDR(0x88000034):

NAME	D31-D0
P_MD_SADDR	MD_SADDR

MD\_SADDR: Recognition start address.

##### In QVGA mode:

If recognition sample type 16x16 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 299$$

If recognition sample type 8x8 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 1199$$

##### In VGA mode:

If recognition sample type 16x16 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 1199$$

If recognition sample type 8x8 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 4799$$

#### 14.6.3 P\_MD\_POS(0x88000038):

NAME	D24-D16	D15-D10	D9-D0
P_MD_POS	MD_VPOS		MD_HPOS

MD\_HPOS: Horizontal IRQ position setting register

MD\_VPOS: Vertical IRQ position setting register

**NOTE:** When the sensor controller detect the current position is equal to {MD\_HPOS, MD\_VPOS}, an interrupt flag will be asserted and the color of this position will be



stored in MD\_RGB register.

#### 14.6.4 P\_MD\_SADDR1(0x8800003C):

NAME	D31-D0
P_MD_SADDR1	MD_SADDR1

MD\_SADDR1: Color Classification start address.

##### In QVGA mode:

If recognition sample type 16x16 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 75$$

If recognition sample type 8x8 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 300$$

##### In VGA mode:

If recognition sample type 16x16 is used, the actual address mapping is:

$$\text{MD\_SADDR} \sim \text{MD\_SADDR} + 300$$

#### 14.6.5 P\_MD\_CTABLE0(0x88000040):

NAME	D31-D0
P_MD_CTABLE0	MD_CTABLE0

MD\_CTABLE0: Color Classification table 0, color of index 0 ~ 15

#### 14.6.6 P\_MD\_CTABLE1(0x88000044):

NAME	D31-D0
P_MD_CTABLE1	MD_CTABLE1

MD\_CTABLE1: Color Classification table 1, color of index 16 ~ 31

#### 14.6.7 P\_MD\_CTABLE2(0x88000048):

NAME	D31-D0
P_MD_CTABLE2	MD_CTABLE2

MD\_CTABLE2: Color Classification table 2, color of index 32 ~ 47

#### 14.6.8 P\_MD\_CTABLE3(0x8800004C):

NAME	D31-D0
P_MD_CTABLE3	MD_CTABLE3

MD\_CTABLE3: Color Classification table 3, color of index 48 ~ 54

#### 14.6.9 P\_MD\_REG1(0x88000050):

NAME	D31-D24	D23-D16	D15-D8	D7-D0
P_MD_REG1	L1	R1	T1	B1

B1: Bottom boundary of region 1, -128~128.

T1: Top boundary of region 1, -128~128.



R1: Right boundary of region 1, -128~128.

T1: Left boundary of region 1, -128~128.

#### 14.6.10 P\_MD\_REG2(0x88000054):

NAME	D31-D24	D23-D16	D15-D8	D7-D0
P_MD_REG2	L2	R2	T2	B2

B2: Bottom boundary of region 2, -128~128.

T2: Top boundary of region 2, -128~128.

R2: Right boundary of region 2, -128~128.

T2: Left boundary of region 2, -128~128.

#### 14.6.11 P\_MD\_REG3(0x88000058):

NAME	D31-D24	D23-D16	D15-D8	D7-D0
P_MD_REG3	L2	R3	T3	B3

B3: Bottom boundary of region 3, -128~128.

T3: Top boundary of region 3, -128~128.

R3: Right boundary of region 3, -128~128.

T3: Left boundary of region 3, -128~128.

#### 14.6.12 P\_MD\_TH(0x8800005C):

NAME	D31-D24	D23-D16	D15-D8	D7-D0
P_MD_TH	TH_B	TH_W	TH1	TH2

TH2: Threshold of region 2, 0~191.

TH1: Threshold of region 1, 0~191.

TH\_W: Threshold of white color, 0~191.

TH\_B: Threshold of black color, 0~191.

#### 14.6.13 P\_MD\_RGB(0x88000074):

NAME	D23-D0
P_MD_RGB	MD_RGB

MD\_RGB: The capture color value.

When YUVOUT is 1, the value in this register is YUV888.

When YUVOUT is 0, the value in this register is RGB888.





## 14.7 Interrupt

Interrupt register 是由 AHB clock 驅動，當某些 event 發生時，硬體會將此 register 變成 1。

CSI 的 interrupt 有 mask register 可設定，當此 register 填 1 時，interrupt 不會傳出 CSI。

CPU 收到 interrupt 會來讀 status register，硬體會利用讀 status register 的 event 來清除 interrupt。

### 14.7.1 P\_CSI\_IRQEN(0x88000078)

NAME	D6	D5	D4	D3	D2	D1	D0
P_CSI_IRQEN	MD_UF_IRQ	FRAME_DIS_IRQ	POS_HIT_IRQ	MD_FRAME_IRQ	FRAME_END_IRQ	TG_CAPACK_IRQ	TG_OVERFLOW_IRQ

TG\_OVERFLOW\_IRQ: TG\_OVERFLOW\_IRQ enable register.

0: Disable TG\_OVERFLOW\_IRQ.

1: Enable TG\_OVERFLOW\_IRQ.

TG\_CAPACK\_IRQ: TG\_CAPACK\_IRQ enable register.

0: Disable TG\_CAPACK\_IRQ.

1: Enable TG\_CAPACK\_IRQ.

FRAME\_END\_IRQ: FRAME\_END\_IRQ enable register.

0: Disable FRAME\_END\_IRQ.

1: Enable FRAME\_END\_IRQ.

MD\_FRAME\_IRQ: MD\_FRAME\_IRQ enable register.

0: Disable MD\_FRAME\_IRQ.

1: Enable MD\_FRAME\_IRQ.

POS\_HIT\_IRQ: POS\_HIT\_IRQ enable register.

0: Disable POS\_HIT\_IRQ.

1: Enable POS\_HIT\_IRQ.

FRAME\_DIS\_IRQ: FRAMD\_DIS\_IRQ enable register.

0: Disable FRAMD\_DIS\_IRQ.

1: Enable FRAMD\_DIS\_IRQ.

MD\_UF\_IRQ: MD\_UF\_IRQ enable register.

0: Disable MD\_UF\_IRQ.

1: Enable MD\_UF\_IRQ.

### 14.7.2 P\_CSI\_IRQSTS(0x8800007C)

NAME	D6	D5	D4	D3	D2	D1	D0
P_CSI_IRQSTS	MD_UF_STS	FRAME_DIS_STS	POS_HIT_STS	MD_FRAME_STS	FRAME_END_STS	TG_CAPACK_STS	TG_OVERFLOW_STS

TG\_OVERFLOW\_STS: TG\_OVERFLOW\_IRQ status register.

Read 0: TG\_OVERFLOW\_IRQ not happen.

Read 1: The sensor output data overflow situation is happened.

Write 0: No effect.

	Write 1: Clear this bit.
TG_CAPACK_IRQ:	TG_CAPACK IRQ status register.
	Read 0: TG_CAPACK IRQ not happen.
	Read 1: The capture procedure is done.
	Write 0: No effect.
	Write 1: Clear this bit.
FRAME_END_IRQ:	FRAME_END IRQ status register.
	Read 0: FRAME_END IRQ not happen.
	Read 1: A frame is end.
	Write 0: No effect.
	Write 1: Clear this bit.
MD_FRAME_IRQ:	MD_FRAME IRQ status register.
	Read 0: MD_FRAME IRQ not happen.
	Read 1: A motion detect frame is end.
	Write 0: No effect.
	Write 1: Clear this bit.
POS_HIT_IRQ:	POS_HIT IRQ status register.
	Read 0: POS_HIT IRQ not happen.
	Read 1: The luster is hit the position defined in {MD_HPOS, MD_VPOS}
	Write 0: No effect.
	Write 1: Clear this bit.
FRAME_DIS_IRQ:	Frame disable interrupt.
	Read 0: FRAME_DIS IRQ not happen.
	Read 1: A frame disable flag is received by CSI module.
	Write 0: No effect.
	Write 1: Clear this bit.
MD_UF_IRQ:	Motion Detect Under Flow interrupt.
	Read 0: MD_UF IRQ not happen.
	Read 1: A motion detect buffer under-run condition is happened.
	Write 0: No effect.
	Write 1: Clear this bit.



## 14.8 Color Mode

### 14.8.1 P\_CSI\_Y2R\_A1(0x880000E8)

NAME	D29-D20	D19-D10	D9-D0
P_CSI_Y2R_A1	Y2R_A11	Y2R_A12	Y2R_A13

Y2R\_A13: A13 parameter of YUV $\Rightarrow$  RGB transfer

Y2R\_A12: A12 parameter of YUV $\Rightarrow$  RGB transfer

Y2R\_A11: A11 parameter of YUV $\Rightarrow$  RGB transfer

### 14.8.2 P\_CSI\_Y2R\_A2(0x880000EC)

NAME	D29-D20	D19-D10	D9-D0
P_CSI_Y2R_A2	Y2R_A21	Y2R_A22	Y2R_A23

Y2R\_A23: A23 parameter of YUV $\Rightarrow$  RGB transfer

Y2R\_A22: A22 parameter of YUV $\Rightarrow$  RGB transfer

Y2R\_A21: A21 parameter of YUV $\Rightarrow$  RGB transfer

### 14.8.3 P\_CSI\_Y2R\_A3(0x880000F0)

NAME	D29-D20	D19-D10	D9-D0
P_CSI_Y2R_A3	Y2R_A31	Y2R_A32	Y2R_A33

Y2R\_A33: A33 parameter of YUV $\Rightarrow$  RGB transfer

Y2R\_A32: A32 parameter of YUV $\Rightarrow$  RGB transfer

Y2R\_A31: A31 parameter of YUV $\Rightarrow$  RGB transfer

**NOTE:** The parameter of YUV $\Rightarrow$  RGB transfer is all 10 bits signed integer. It will be divided by 256 in the hardware. In other word, the value of these parameters have the following range.

$$-512/256 \leq Y2R\_Axx \leq 511/256$$

### 14.8.4 P\_CSI\_R2Y\_A1(0x880000E8)

NAME	D29-D20	D19-D10	D9-D0
P_CSI_R2Y_A1	R2Y_A11	R2Y_A12	R2Y_A13

R2Y\_A13: A13 parameter of RGB $\Rightarrow$  YUV transfer

R2Y\_A12: A12 parameter of RGB $\Rightarrow$  YUV transfer

R2Y\_A11: A11 parameter of RGB $\Rightarrow$  YUV transfer

### 14.8.5 P\_CSI\_R2Y\_A2(0x880000EC)

NAME	D29-D20	D19-D10	D9-D0
P_CSI_R2Y_A2	R2Y_A21	R2Y_A22	R2Y_A23

R2Y\_A23: A23 parameter of RGB $\Rightarrow$  YUV transfer

R2Y\_A22: A22 parameter of RGB $\Rightarrow$  YUV transfer

R2Y\_A21: A21 parameter of RGB $\Rightarrow$  YUV transfer

#### 14.8.6 P\_CSI\_R2Y\_A3(0x880000F0)

NAME	D29-D20	D19-D10	D9-D0
P_CSI_R2Y_A3	R2Y_A31	R2Y_A32	R2Y_A33

R2Y\_A33: A33 parameter of RGB $\Rightarrow$  YUV transfer

R2Y\_A32: A32 parameter of RGB $\Rightarrow$  YUV transfer

R2Y\_A31: A31 parameter of RGB $\Rightarrow$  YUV transfer

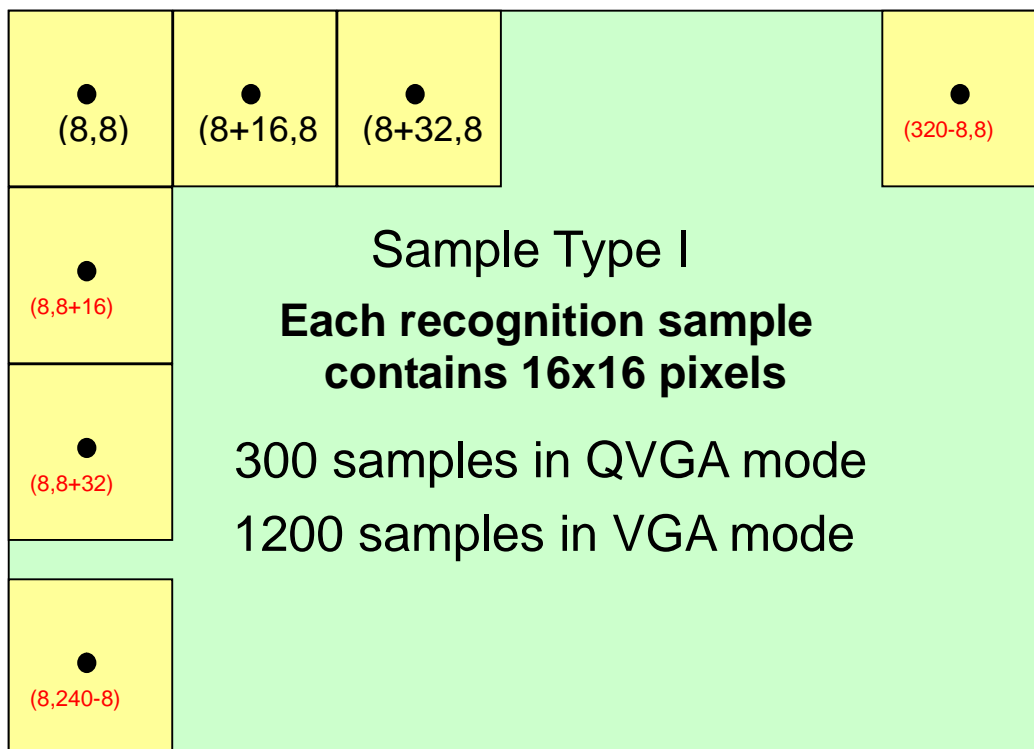
**NOTE:** The parameter of RGB $\Rightarrow$  YUV transfer is all 9 bits signed integer. It will be divided by 256 in the hardware. In other word, the values of these parameters have the following range.

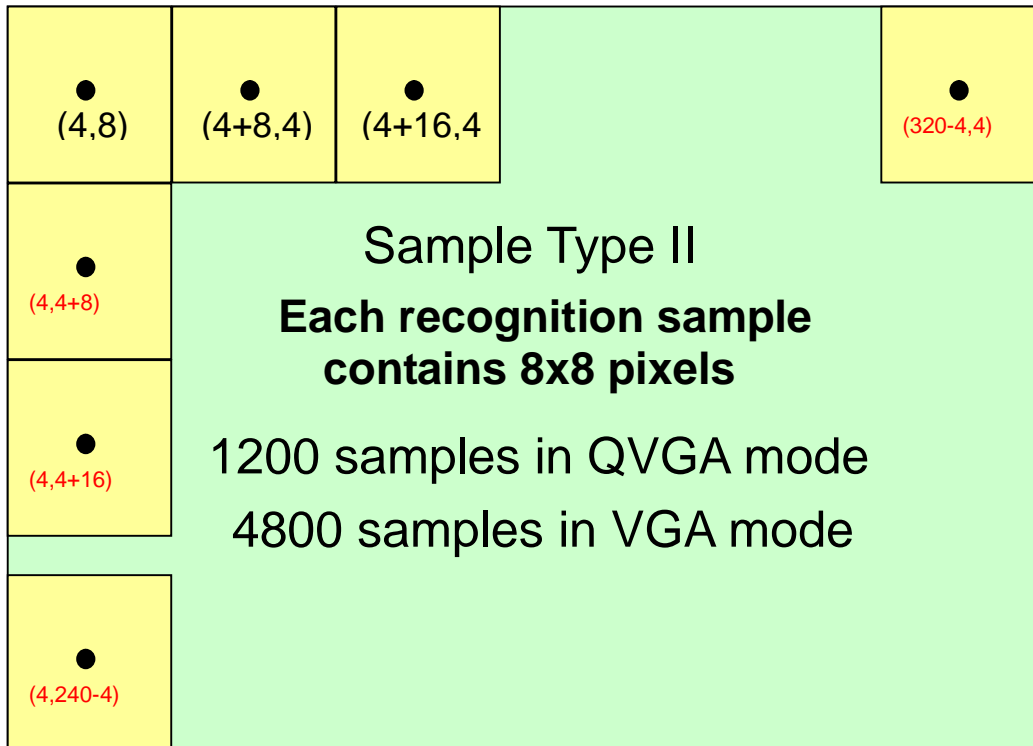
$$-256/256 \leq R2Y\_Axx \leq 255/256$$

**CSI\_Y2R and CSI\_R2Y share the same register.** When YUVIN mode is selected, these registers must be filled by YUV $\Rightarrow$ RGB parameter. When RGBIN mode is selected, these registers must be filled by RGB $\Rightarrow$ YUV parameter.

### 14.9 Motion detection

#### 14.9.1 Recognition Sample Type





#### 14.9.2 Recognition Reference Filed / Frame

If CPU does not have much time to handle motion detect algorithm, recognition data can not always calculated at each frame. The following table shows the reference field or frame decided by setting up parameters. For examples, if INTERLACE = 1 and RC\_FIELD = 0 and FRAME\_TYPE = 2, each reference frame is frame2 field0 instead of frame2 field1.

INTERLACE	RC_FIELD	FRAME_TYPE	Field0	Field1	Frame0	Frame1	Frame2	Frame3
0	0/1	0			HIT	HIT	HIT	HIT
0	0/1	1				HIT		HIT
0	0/1	2					HIT	
0	0/1	3						HIT
1	0	0	REF		HIT	HIT	HIT	HIT
1	0	1	REF			HIT		HIT
1	0	2	REF				HIT	
1	0	3	REF					HIT
1	1	0		REF	HIT	HIT	HIT	HIT
1	1	1		REF		HIT		HIT
1	1	2		REF			HIT	
1	1	3		REF				HIT

**Notes:** REF means which field is referred.



FRAME_TYPE	FRAME_COUNTER
0	Always 0
1	Always from 0 to 1
2	Always from 0 to 2
3	Always from 0 to 3



---

---

## 15 CCIR601/656 VIDEO INPUT

---

---

## 16 USB 1.1

當 USB module 啟動時，USB module 吃的 APB clock 是 24MHz。

In non-DMA mode =>寫到 USB device 的 data 以及從 USB device 讀出的 data，由 CPU 填入/讀出 USB module。

In DMA mode =>寫到 USB device 的 data 以及從 USB device 讀出的 data，由 APBDMA 填入/讀出 USB module。

USB 輸出給 DMA 的 RD\_REQ (level, high-active) 拉起時代表有一筆 16-bit 的 data 等待被 APBDMA 讀走。

USB 輸出給 DMA 的 WR\_REQ (level, high-active) 拉起時代表有一筆 16-bit 的 data 等待由 APBDMA 寫入。

APBDMA 永遠都看得見上述的 RD\_REQ, WR\_REQ 訊號 (不論有無使用 DMA)。

USB module 接到 interrupt controller 的 IRQ 訊號可被設定為 masked。此 IRQ 由 APB clock 驅動。CPU 清除 USB 內部的 interrupt flag 即可清除此 IRQ。

CPU 必須設定 DMA channel 對 DRAM 的 start address 和 end address，當 APBDMA 對 MIU 發出的 HADDR 等於 end address 且完成該次 transfer 之後，DMA 會發出 IRQ 給 interrupt controller (通知 CPU)，此 IRQ 訊號可被設定為 masked。

## USB's Memory Map

				CPU address[15:0]	APB addr. to USB
3	2	1	0	0	0
7	6	5	4	4	1
11	10	9	8	8	2
15	14	13	12	c	3

※每個小方塊代表memory space中的一個byte，  
所以同一行的四個方塊代表一個32-bit word，  
灰色方塊代表該處一定不會用到

※USB收到的APB address其實就是CPU address[15:2]  
每個APB address代表一個16-bit word



### 16.1.1 USB Device

When SPG290 is used as USB Device, it supports 4 endpoints, control pipe, bulk in, bulk out, and interrupt in. For control pipe, when SPG290 receives the standard command, it will auto reply it except Get / Set Descriptor. That is, when it received GET\_STATUS, CLEAR\_FEATURE, SET\_FEATURE, SET\_ADDRESS, GET\_CONFIGURATION, SET\_CONFIGURATION, GET\_INTERFACE, and SET\_INTERFACE, the SPG290 will auto reply these standard commands. For bulk in and bulk out, the maximum packet size is 64 bytes. SPG290 supports non-DMA or DMA transmit / receive. For non-DMA mode, it is 8 bits for MCU access. And it is 16 bits access for DMA mode.

### 16.1.2 USB Mini Host

SPG290 can be used as the USB mini host. It supports commands, IN, and OUT transfer. For command transfer, the maximum packet size is 64 bytes. For IN and OUT transfer, the maximum packet size is 64 bytes, and it is 8 bits for MCU access or 16 bits for DMA access. When it uses DMA mode, the data to be transferred must be multi-pipe of 64 bytes.

### 16.1.3 Control Register

USB Device Register Summary Table

Name	Address	Description
P_USBD_Config	0x881B00C0	USB Configuration Register
P_USBD_Device	0x881B015C	USB Device Register
P_USBD_Function	0x881B00C4	USB Function Register
P_USBD_DMAINT	0x881B0164	USB DMA Interrupt Register
P_USBD_PMR	0x881B00C8	USB Power Management Register
P_USBD_EP0Data	0x881B00CC	USB Endpoint0 Data Register
P_USBD_BIData	0x881B00D0	USB Bulk In Data Register
P_USBD_BOData	0x881B00D4	USB Bulk Out Data Register
P_USBD_INTINData	0x881B00D8	USB Interrupt In Data Register
P_USBD_NullPkt	0x881B0160	USB Null Packet Register
P_USBD_EPEvent	0x881B00DC	USB Endpoint Event Register
P_USBD_GLOINT	0x881B00E0	USB Global Interrupt Register
P_USBD_INTEN	0x881B00E4	USB Interrupt Enable Register
P_USBD_INT	0x881B00E8	USB Interrupt Register
P_USBD_SCI NTEN	0x881B00EC	USB Standard Command Interrupt Enable Register
P_USBD_SCINT	0x881B00F0	USB Standard Command Interrupt Register



P_USBD_EPAutoSet	0x881B00F4	USB Endpoint Auto Set Register
P_USBD_EPSetStall	0x881B00F8	USB Endpoint Set Stall Register
P_USBD_EPBufClear	0x881B00FC	USB Endpoint Buffer Clear Register
P_USBD_EPEvntClear	0x881B0100	USB Endpoint Event Clear Register
P_USBD_EP0WrtCount	0x881B0104	USB Endpoint0 Write Count Register
P_USBD_BOWrtCount	0x881B0108	USB Bulk Out Write Count Register
P_USBD_EP0BufPointer	0x881B010C	USB Endpoint0 Buffer Pointer Register
P_USBD_BIBufPointer	0x881B0110	USB Bulk In Buffer Pointer Register
P_USBD_BOBufPointer	0x881B0114	USB Bulk Out Buffer Pointer Register
P_USBD_EP0RTR	0x881B0118	USB Endpoint0 bmRequestType Register
P_USBD_EP0RR	0x881B011C	USB Endpoint0 bRequest Register
P_USBD_EP0VR	0x881B0120	USB Endpoint0 wValue Register
P_USBD_EP0IR	0x881B0124	USB Endpoint0 wIndex Register
P_USBD_EP0LR	0x881B0128	USB Endpoint0 wLength Register
P_USBD_DMAWrtCount	0x881B0140	USB DMA Byte Count Register
P_USBD_DMAACKCount	0x881B0144	USB DMA ACK Count Register
P_USBD_EPStall	0x881B0150	USB Endpoint Stall Bit Register
P_USBD_TNSP	0x881B017C	USB transceiver Parameter Register

USB Host Register Summary Table

Name	Address	Description
P_USBH_Config	0x881C0000	USB Host Configuration Register
P_USBH_TimeConfig	0x881C0004	USB Host Timing Configuration Register
P_USBH_Data	0x881C0008	USB Host Data Register
P_USBH_Transfer	0x881C000C	USB Host Transfer Register
P_USBH_DveAddr	0x881C0010	USB Device Address Register
P_USBH_DveEP	0x881C0014	USB Device Endpoint Register
P_USBH_TXCount	0x881C0018	USB Host Transmit Count Register
P_USBH_RXCount	0x881C001C	USB Receive Count Register
P_USBH_FIFOInPointer	0x881C0020	USB Host FIFO Input Pointer Register
P_USBH_FIFOOuPointer	0x881C0024	USB Host FIFO Output Pointer Register
P_USBH_AutoInByteCount	0x881C0028	USB Host Automatic In Transaction Byte Count Register
P_USBH_AutoOutByteCount	0x881C002C	USB Host Automatic Out Transaction Byte Count Register
P_USBH_AutoTrans	0x881C0030	USB Host Auto Transfer Register



P_USBH_Status	0x881C0034	USB Host Status Register
P_USBH_INT	0x881C0038	USB Host Interrupt Register
P_USBH_INTEN	0x881C003C	USB Host Interrupt Enable Register
P_USBH_StorageRST	0x881C0040	USB Storage Reset Register
P_USBH_SoftRST	0x881C0044	USB Software Reset Register / Device Plug Out Register
P_USBH_SOFTimer	0x881C0048	USB SOF Timer Register
P_USBH_FrameNum	0x881C004C	USB Frame Number Register
P_USBH_OTGConfig	0x881C0050	USB OTG Configuration Register
P_USBH_VBusSet	0x881C0054	USB VBUS Set Register
P_USBH_VbusStatus	0x881C0058	USB VBUS Status Register
P_USBH_INAckCount	0x881C005C	USB IN ACK Count Register
P_USBH_OutAckCount	0x881C0060	USB OUT ACK Count Register
P_USBH_RSTAckCount	0x881C0064	USB Reset ACK Count Register
P_USBH_Storage1/2	0x881C0068	For Debugging
P_USBH_DReadback	0x881C006C	USB D+ / D- Readback Register

#### USB Device Register Definition

##### P\_USBD\_Config 0x881B00C0 USB Configuration Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	RWUPEN	SPWR	USBEN	TNSPL	TNSPH	BYPASS
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:12]			Reserved	
11:8	TNSPH	R/W	USB Transceiver Pull High	
7:6	TNSPL	R/W	USB Transceiver Pull Low	
5	RWUPEN	R/W	Remote Wakeup Enable If this bit is set to "1", the remote wakeup is supported. Or, it is not supported.	0= Remote Wakeup is not supported 1= Remote Wakeup is supported
4	SPWR	R/W	Self Power of Device This bit is used to indicate if USB device is self-powered or not.	0= USB device is bus-powered 1= USB device is self-powered
3	USBEN	R/W	USB Transceiver Enable Write "1" to this bit to enable the USB transceiver.	
[2:1]			Reserved	



0	BYPASS	R/W	USB Bypass Mode	0= Bypass disable 1= Bypass enable, the inner USB transceiver is disabled.
---	--------	-----	-----------------	---

P_USBD_Device		0x881B015C								USB Device Register											
Bit		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Function		EP4_Type		EP3_Type		EP2_Type		EP1_Type		EP4_IO		EP3_IO		EP2_IO		EP1_IO		-	-	-	MOD
Default		0	1	1	1	1	0	1	0	0	1	0	1	0	0	0	0	0			

Bit	Function	Type	Description	Condition
[15:14] ]	EP4_Type	R/W	Endpoint4 Type These two bits are used to indicate the type of endpoint4	00 = Reserved 01 = Reserved 10 = Bulk 11 = Interrupt
[13:12] ]	EP3_Type	R/W	Endpoint3 Type These two bits are used to indicate the type of endpoint3	00 = Reserved 01 = Reserved 10 = Bulk 11 = Interrupt
[11:10] ]	EP2_Type	R/W	Endpoint2 Type These two bits are used to indicate the type of endpoint2	00 = Reserved 01 = Reserved 10 = Bulk 11 = Interrupt
[9:8]	EP1_Type	R/W	Endpoint1 Type These two bits are used to indicate the type of endpoint1	00 = Reserved 01 = Reserved 10 = Bulk 11 = Interrupt
7	EP4_IO	R/W	Endpoint4 IN/OUT This bit is used to indicate the EP4 is in or out.	0 = OUT 1 = IN
6	EP3_IO	R/W	Endpoint3 IN/OUT This bit is used to indicate the EP3 is in or out.	0 = OUT 1 = IN
5	EP2_IO	R/W	Endpoint2 IN/OUT This bit is used to indicate the EP2 is in or out.	0 = OUT 1 = IN
4	EP1_IO	R/W	Endpoint1 IN/OUT This bit is used to indicate the EP1 is in or out.	0 = OUT 1 = IN
[3:0]			Reserved	
1	MODE	R/W	Mode selection	0 = Normal mode 1 = Debug mode

P_USBD_Function					0x881B00C4				USB Function Register							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	SRST	DMA_BOEN	DMA_BIEN	Config_Value		FNC_Addr						
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	Function	Type	Description	Condition
[15:12]			Reserved	
11	SRST	W	Software Reset	
10	DMA_BOEN	R/W	DMA Bulk OUT Enable Write "1" to this to enable the DMA function with bulk out. When this bit is set to 1, the DMA_BIEN must be 0. It indicated Bulk IN must disable when Bulk out is enable in DMA mode.	Write 0= Diabale DMA function with bulk out Write 1= Enable DMA function with bulk out
9	DMA_BIEN	R/W	DMA Bulk IN Enable Write "1" to this to enable the DMA function with bulk in. When this bit is set to 1, the DMA_BOEN must be 0. It indicated Bulk OUT must disable when Bulk in is enable in DMA mode.	Write 0= Diabale DMA function with bulk in Write 1= Enable DMA function with bulk in
[8:7]	Config_Value	R	Configure Value The USB configuration value of the device can be read from these two bits when received set configuration command.	
[6:0]	FNC_Addr	R	Function Address When the device gets " Set Address" command from the host, the address is stored in these bits.	

**P\_USBD\_DMAINT 0x881B0164 USB DMA Interrupt Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DMAINTEN	DMAINT
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:12]			Reserved	
1	DMAINTEN	R/W	DMA Interrupt Enable If this bit is set to "1", hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
0	DMAINT	R/W	Bulk Out Interrupt This bit is set if one of DMA interrupt happened, which indicates Bulk Out or Bulk In transaction is finished in DMA mode. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred

**P\_USBD\_PMR 0x881B00C8 USB Power Management Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	RESWKE	RE_WA	RE_WAFEA	RST	SUS_Mod



Default 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bit	Function	Type	Description	Condition
[15:5]			Reserved	
4	RESWKE	R/W	Resume Wakeup Enable Write "1" to enable "Resume Wakeup". (Returning from suspend mode to indicate system or CPU that USB is wakeup now). Write "0" to disable this function.	
3	RE_WA	R/W	Remote_Wakeup The USB generates resume signaling for a duration of 0.24 micro-second while this bit is set in suspend mode and the USB device has already gotten the SET_FEATURE command of REMOTE_WAKEUP. This bit must be cleared by manual after SUSPEND_MODE is cleared.	
2	RE_WAFE A	R	Remote Wakeup Feature When the device gets SET_FEATURE command of REMOTE_WAKEUP, this bit will set to 1 by hardware. When the device gets CLEAR_FEATURE command of REMOTE_WAKEUP, this bit will clear to 0 by hardware.	
1	RST	R	Reset This bit is assigned to USBBUS reset. 1 means USB reset is high, and 0 means USB reset is low.	
0	SUS_Mod	R	Suspend Mode This bit is set by hardware when it enters suspend mode. This bit is cleared by hardware when it returns from suspend mode or the USB reset signal is generated.	

**P\_USBD\_EP0Data 0x881B00CC USB Endpoint0 Data Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function																
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	EP0DATA	R/W	Endpoint0 Data	

**P\_USBD\_BIData 0x881B00D0 USB Bulk IN Data Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---



Function	-	-	-	-	-	-	-	-	BIDATA							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	BIDATA	R/W	Bulk In Data	

**P\_USBD\_BOData      0x881B00D4      USB Bulk OUT Data Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	BODATA							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	BODATA	R/W	Bulk Out Data	

**P\_USBD\_INTINData      0x881B00D8      USB Interrupt IN Data Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function									INTINDATA							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	INTINDATA	R/W	Interrupt IN Data	

**P\_USBD\_NullPkt      0x881B0160      USB Null Packet Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-				NULLPKT
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:1]			Reserved	
0	NULLPKT	R/W	Write "1" to enable that USB device can send null packet. Write "0" to disable that function	

**P\_USBD\_EPEvent      0x881B00DC      USB Endpoint Event Register**



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	IINNA	IINPR	BONA	BOPR	BOPE	BINA	BIPC	BIPR	EOSNA	E0SEN	E0INNA	E0INPR	E0ONA	E0OPR	E0OPE	E0SPR
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
15	IINNA	R/W	Interrupt IN NACK This bit is set if an IN request happened but the device sent NAK. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
14	IINPR	R/W	Interrupt IN Packet Ready Set this bit to indicate IN packet is ready in the INTERRUPT_IN FIFO. It's set automatically if there are 8 bytes IN data and P_USBD_EPAutoSet[4] is 1. After this bit is set to "1", the hardware will send the data to the Host. This bit is cleared by hardware when the data transaction is finished.	Write 1= Packet is ready.
13	BONA	R/W	Bulk Out NACK This bit is set if an OUT packet happened but the device sent NAK. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
12	BOPR	R/W	Bulk Out Packet Ready This bit is set if an OUT packet is loaded into the bulk out FIFO. Writing 1 to clear the bit or it's cleared automatically if the OUT packet is read from MCU and P_USBD_EPAutoSet[3] is 1.	Read 1= Ready Read 0= Not ready Write 0= No effect Write 1= Clear the flag
11	BOPE	R/W	Bulk Out Packen Enable Write "1" to this bit will enable the incoming packet for OUT data. This bit is automatically cleared after the packet is loaded to bulk out FIFO (BOPR is set to "1"). Write "1" to P_USBD_EPEvntClear[4] will clear this bit.	Write 1= Enable
10	BINA	R/W	Bulk IN NACK This bit is set if an IN request happened but the device sent NAK. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
9	BIPC	R/W	Bulk IN Packet Clear This bit is set if an IN packet is read from the host. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
8	BIPR	R/W	Bulk IN Packet Ready Set this bit to indicate IN packet is ready in the BULK_IN FIFO. It's set automatically if MCU wrote 64 bytes data and P_USBD_EPAutoSet[2] is 1. After this bit is set to "1", the hardware will send the data to the Host. This bit is cleared by hardware when the data transaction is finished.	Write 1= Packet is ready.





7	E0SNA	R/W	EP0 Status NACK This bit is set if the request of status transaction happened but the device sent NAK. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
6	E0SEN	R/W	EP0 Status Enable This bit is set to enable the transaction in status stage. It's automatically cleared if the status stage is finished. Besides, It's set automatically if the status stage is finished and P_USBD_EPAutoSet[5] is 1.	Write 1= Enable
5	E0INNA	R/W	EP0 IN NACK This bit is set if an IN request happened but the device sent NAK. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
4	E0INPR	R/W	EP0 IN Packet Ready Set this bit to indicate IN packet is ready in the Endpoint0 FIFO. It's set automatically if there are 8 bytes IN data and P_USBD_EPAutoSet[1] is 1. After this bit is set to "1", the hardware will send the data to the Host. This bit is cleared by hardware when the data transaction is finished.	Write 1 to indicate IN packet is ready in Endpoint0 FIFO.
3	E0ONA	R/W	EP0 Out NACK This bit is set if an OUT packet happened but the device sent NAK. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 0= No effect Write 1= Clear the flag
2	E0OPR	R/W	EP0 Out Packet Ready This bit is set if an OUT packet is loaded into the endpoint0 FIFO. Writing 1 to clear the bit or it's cleared automatically if the OUT packet is read from MCU and P_USBD_EPAutoSet[0] is 1.	Read 1= Ready Read 0= Not ready Write 0= No effect Write 1= Clear the flag
1	E0OPE	R/W	EP0 Out Packet Enable Write "1" to this bit will enable the incoming packet for OUT data. This bit is automatically cleared after the packet is loaded to endpoint0 FIFO (E0OPR is set to "1"). Write "1" to P_USBD_EPEvntClear[0] will clear this bit.	Write 1= Enable
0	E0SPR	R/W	EP0 Setup Packet Ready This bit is set if a non-standard setup command or get/set descriptor command is loaded into the endpoint0 FIFO. Writing 1 to clear the bit.	Read 1= Occurred Read 0= Not occurred Write 1= Clear Write 0= No effect

P\_USBD\_GLOINT

0x881B00E0

USB Global Interrupt Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	DMA	STANDARD	POWER	INT	BO	BI	EP0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	Function	Type	Description	Condition
[15:7]			Reserved	
6	DMA	R	DMA Interrupt This bit is set if one of DMA interrupt happened. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred
5	STANDARD	R	Standard Command Interrupt This bit is set if one of standard command interrupt happened, except GET/SET Descriptor. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred
4	POWER	R	Power Management Interrupt This bit is set if one of power management interrupt happened. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred
3	INT	R	Interrupt In interrupt This bit is set if one of INTERRUPT_IN interrupt happened. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred
2	BO	R	Bulk Out Interrupt This bit is set if one of BULK_OUT interrupt happened. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred
1	BI	R	Bulk In Interrupt This bit is set if one of BULK_IN interrupt happened. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred
0	EP0	R	Endpoint0 Interrupt This bit is set if one of endpoint0 interrupt happened. Writing 1 to clear the specific interrupt happened would clear this bit.	Read 1= Occurred Read 0= Not occurred

#### P\_USBD\_INTEN

0x881B00E4

#### USB Interrupt Enable Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	RST	RME	SUS	IINNA	IINPC	BONA	BOPS	BINA	BIPC	E0SNA	E0SC	E0INNA	E0INPC	E0ONA	E0OPS	E0SPS
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
15	RST	R/W	Rest Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
14	RME	R/W	Resume Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable



13	SUS	R/W	Suspend Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
12	IINNA	R/W	Interrupt In NACK Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
11	IINPC	R/W	Interrupt In Packet Clear Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
10	BONA	R/W	Bulk Out NACK Interrupt Enable. If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
9	BOPS	R/W	Bulk Out Packet Set Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
8	BINA	R/W	Bulk In NACK Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
7	BIPC	R/W	Bulk In Packet Clear Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
6	E0SNA	R/W	EP0 Status NACK Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
5	E0SC	R/W	EP0 Status Clear Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
4	E0INNA	R/W	EP0 In NACK Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable



3	E0INPC	R/W	EP0 In Packet Clear Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
2	E0ONA	R/W	EP0 Out NACK Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
1	E0OPS	R/W	EP0 Out Packet Set Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
0	E0SPS	R/W	EP0 Setup Packet Set Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable

**P\_USBD\_INT**

**0x881B00E8**

**USB Interrupt Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	RST	RME	SUS	IINNA	IINPC	BONA	BOPS	BINA	BIPC	E0SNA	E0SC	E0INNA	E0INPC	E0ONA	E0OPS	E0SPS
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
15	RST	R/W	Reset Interrupt Flag The interrupt is set if the enable bit is 1 and the device got USB RESET.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
14	RME	R/W	Resume Interrupt Flag The interrupt is set if the enable bit is 1 and the device got RESUME in SUSPEND state.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
13	SUS	R/W	Suspend Interrupt Flag The interrupt is set if the enable bit is 1 and the device got SUSPEND.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
12	IINNA	R/W	Interrupt In NACK Interrupt Flag This bit is set if the enable bit is 1 and an IN request happened but the device sent NAK.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
11	IINPC	R/W	Interrupt In Packet Clear Interrupt Flag This bit is set if the enable bit is 1 and an IN packet is read from the host.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag



10	BONA	R/W	Bulk Out NACK Interrupt Flag This bit is set if the enable bit is 1 and an OUT packet happened but the device sent NAK.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
9	BOPS	R/W	Bulk Out Packet Set Interrupt Flag This bit is set if the enable bit is 1 and an OUT packet is loaded into the endpoint0 FIFO.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
8	BINA	R/W	Bulk In NACK Interrupt Flag This bit is set if the enable bit is 1 and an IN request happened but the device sent NAK.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
7	BIPC	R/W	Bulk In Packet Clear Interrupt Flag This bit is set if the enable bit is 1 and an IN packet is read from the host.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
6	E0SNA	R/W	EP0 Status NACK Interrupt Flag This bit is set if the enable bit is 1 and the request of status transaction happened but the device sent NAK.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
5	E0SC	R/W	EP0 Status Clear Interrupt Flag This bit is set if the enable bit is 1 and the status stage is finished.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
4	E0INNA	R/W	EP0 In NACK Interrupt Flag This bit is set if the enable bit is 1 and an IN request happened but the device sent NAK.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
3	E0INPC	R/W	EP0 In Packet Clear Interrupt Flag This bit is set if the enable bit is 1 and an IN packet is read from the host.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
2	E0ONA	R/W	EP0 Out NACK Interrupt Flag This bit is set if the enable bit is 1 and an OUT packet happened but the device sent NAK.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
1	E0OPS	R/W	EP0 Out Packet Set Interrupt Flag This bit is set if the enable bit is 1 and an OUT packet is loaded into the endpoint0 FIFO.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
0	E0SPS	R/W	EP0 Setup Packet Set Interrupt Flag The interrupt is set if the enable bit is 1 and a non-standard setup command or get/set descriptor command is loaded into the endpoint0 FIFO.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag

**P\_USBD\_SCINTEN      0x881B00EC      USB Standard Command Interrupt Enable Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	GSTS	CFEA	SFEA	SADD	GCON	SCON	GINT	SINT
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	Function	Type	Description	Condition
[15:8]			Reserved	
7	GSTS	R/W	Get Status Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
6	CFEA	R/W	Clear Feature Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
5	SFEA	R/W	Set Feature Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
4	SADD	R/W	Set Address Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked. To select between IRQ3 and FIQ, please refer to <b>Chapter Interrupt</b>	0= Disable 1= Enable
3	GCON	R/W	Get Configuration Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
2	SCON	R/W	Set Configuration Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
1	GINT	R/W	Get Interface Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable
0	SINT	R/W	Set Interface Interrupt Enable If this bit is set to "1" and interrupt occurs, hardware will issue an IRQ to CPU. If this bit is cleared to "0", this interrupt will be masked.	0= Disable 1= Enable

P_USBD_SCINT			0x881B00F0						USB Standard Command Interrupt Register							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	GSTS	CFEA	SFEA	SADD	GCON	SCON	GINT	SINT
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	Function	Type	Description	Condition
[15:8]			Reserved	
7	GSTS	R/W	Get Status Interrupt Flag The interrupt is set if the enable bit is 1 and GET_STATUS command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
6	CFEA	R/W	Clear Feature Interrupt Flag The interrupt is set if the enable bit is 1 and CLEAR_FEATURE command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
5	SFEA	R/W	Set Feature Interrupt Flag The interrupt is set if the enable bit is 1 and SET_FEATURE command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
4	SADD	R/W	Set Address Interrupt Flag The interrupt is set if the enable bit is 1 and SET_ADDRESS command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
3	GCON	R/W	Get Configuration Interrupt Flag The interrupt is set if the enable bit is 1 and GET_CONFIGURATION command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
2	SCON	R/W	Set Configuration Interrupt Flag The interrupt is set if the enable bit is 1 and SET_CONFIGURATION command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
1	GINT	R/W	Get Interface Interrupt Flag The interrupt is set if the enable bit is 1 and GET_INTERFACE command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
0	SINT	R/W	Set Interface Interrupt Flag The interrupt is set if the enable bit is 1 and SET_INTERFACE command happened.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag

**P\_USBD\_EPAutoSet 0x881B00F4 USB Endpoint Auto Set Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	EP0ASE	IAINPR	BAOPE	BAIPR	E0AIPR	E0AOPE
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:6]			Reserved	
5	EP0ASE	R/W	EP0 Auto Status Enable Set Please refer to P_USBD_EPEvent[6]	
4	IAINPR	R/W	Interrupt Auto In Packet Ready Set Please refer to P_USBD_EPEvent[14]	
3	BAOPE	R/W	Bulk Auto Out Packet Enable Set Please refer to P_USBD_EPEvent[11] or [12]	



2	BAIPR	R/W	Bulk Auto In Packet Ready Set Please refer to P_USBD_EPEvent[8]	
1	E0AIPR	R/W	EP0 Auto In Packet Ready Set Please refer to P_USBD_EPEvent[4]	
0	E0AOPE	R/W	EP0 Auto Out Packet Enable Set Please refer to P_USBD_EPEvent[1] or [2]	

P_USBD EPSetStall		0x881B00F8								USB Endpoint Set Stall Register							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Function	-	-	-	-	-	-	-	-	-	-	-	-	IINS	BOS	BIS	EP0S	
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	Function	Type	Description	Condition
[15:4]			Reserved	
3	IINS	R/W	Interrupt In Set Stall	1 = enable STALL response 0 = disable STALL response.
2	BOS	R/W	Bulk Out Set Stall	1 = enable STALL response 0 = disable STALL response.
1	BIS	R/W	Bulk In Set Stall	1 = enable STALL response 0 = disable STALL response.
0	EP0S	R/W	EP0 Set Stall Write 1 to this bit to generate STALL if host request happened. This bit is automatically cleared if one of SETUP command is loaded into endpoint0 FIFO.	1 = enable STALL response 0 = disable STALL response.

P_USBD_EPBufClear			0x881B00FC						USB Endpoint Buffer Clear Register							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	IINBC	BOBC	BIBC	EP0BC
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:4]			Reserved	
3	IINBC	W	Interrupt In Buffer Clear Write "1" to clear Interrupt In buffer	
2	BOBC	W	Bulk Out Buffer Clear Write "1" to clear Bulk Out buffer	
1	BIBC	W	Bulk In Buffer Clear Write "1" to clear Bulk In buffer	
0	EP0BC	W	EP0 Buffer Clear Write "1" to clear EP0 buffer	

P_USBD_EPEvntClear	0x881B0100	USB Endpoint Event Clear Register
--------------------	------------	-----------------------------------





Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	IINPC	BOEC	BIPC	EP0SC	EP0IPC	EP0OEC
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:6]			Reserved	
5	IINPC	W	Interrupt In Packet Clear Write "1" to this bit to clear P_USBD_EPEvent[14]	
4	BOEC	W	Bulk Out Enable Clear Write "1" to this bit to clear P_USBD_EPEvent[11]	
3	BIPC	W	Bulk In Packet Clear Write "1" to this bit to clear P_USBD_EPEvent[9]	
2	EP0SC	W	EP0 Status Clear Write "1" to this bit to clear P_USBD_EPEvent[6]	
1	EP0IPC	W	EP0 In Packet Clear Write "1" to this bit to clear P_USBD_EPEvent[4]	
0	EP0OEC	W	EP0 Out Enable Clear Write "1" to this bit to clear P_USBD_EPEvent[1]	

**P\_USBD\_EP0WrtCount 0x881B0104 USB Endpoint0 Write Count Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	EP0WC			
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:4]			Reserved	
[3:0]	EP0WC	R	EP0 Write Count Read these bits to indicate the number of data in the EP0 FIFO.	

**P\_USBD\_BOWrtCount 0x881B0108 USB Bulk OUT Write Count Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	BOWC						
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:7]			Reserved	



[6:0]	BOWC	R	Bulk Out Write Count Read these bits to indicate the number of data in the Bulk Out FIFO.	
-------	------	---	--	--

**P\_USBD\_EP0BufPointer 0x881B010C USB Endpoint0 Buffer Pointer Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	EP0WBP			EP0RBP		
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:6]			Reserved	
[5:3]	EP0WBP	R	EP0 Write Buffer Pointer The write pointer of EP0 FIFO	
[2:0]	EP0RBP	R	EP0 Read Buffer Pointer The read pointer of EP0 FIFO	

**P\_USBD\_BIBufPointer 0x881B0110 USB Bulk IN Buffer Pointer Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	BIBRP								BIBWP							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]	BIBRP		Bulk IN Buffer Write Pointer The write pointer of Bulk IN FIFO	
[7:0]	BIBWP	R	Bulk IN Buffer Read Pointer The read pointer of Bulk IN FIFO	

**P\_USBD\_BOBufPointer 0x881B0114 USB Bulk OUT Buffer Pointer Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	BOBRP								BOBWP							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]	BOBRP		Bulk OUT Buffer Write Pointer The write pointer of Bulk OUT FIFO	
[7:0]	BOBWP	R	Bulk OUT Buffer Read Pointer The read pointer of Bulk OUT FIFO	

**P\_USBD\_EP0TR 0x881B0118 USB Endpoint0 bmRequestType Register**



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	EP0RTR							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	EP0RTR	R	EP0 bmRequestType The bmRequestType of setup command	

**P\_USBD\_EP0RR                      0x881B011C                      USB Endpoint0 bRequest Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	EP0RR							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	EP0RR	R	EP0 bRequest The bRequest of setup command	

**P\_USBD\_EP0VR                      0x881B0120                      USB Endpoint0 wValue Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	EP0VR															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:0]	EP0VR	R	EP0 wValue The wValue of setup command	

**P\_USBD\_EP0IR                      0x881B0124                      USB wIndex Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	EP0IR															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:0]	EP0IR	R	EP0 wIndex The wIndex of setup command	

**P\_USBD\_EP0LR                      0x881B0128                      USB Endpoint0 wLength Register**



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	EP0LR															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:0]	EP0LR	R	EP0 wLength The wLength of setup command	

**P\_USBD\_DMAWrtCount 0x881B0140 USB DMA Byte Count Register**

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	DMAWC																							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[23:0]	DMAWC	R/W	DMA Write Count The register is used in DMA mode. Setting this register to indicate how many word data is transferred.	

**P\_USBD\_DMAACK 0x881B0144 USB DMA ACK Register**

Bit	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	DMAACK																		
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[18:0]	DMAACK	R/W	DMA ACK Count The register is used in DMA mode. It indicates how many transactions (each transaction means a 64-bytes packet transferring) are not finished yet.	Write this port to reset DMAWC and DMAACK

**P\_USBD\_EPStall 0x881B0150 USB Endpoint Stall Bit Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	IISB	BOS	BISS	EP0	-	-	-	-	IISB	BOS	BISB	EP0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
11	EP0SS	R/W	Interrupt IN It indicates that USB device has replied the request by sending STALL	Write "1" to clear this bit



10	BOSS	R/W	Bulk OUT It indicates that USB device has replied the request by sending STALL	Write "1" to clear this bit
9	BISS	R/W	Bulk IN It indicates that USB device has replied the request by sending STALL	Write "1" to clear this bit
8	EP0SS	R/W	Endpoint0 It indicates that USB device has replied the request by sending STALL	Write "1" to clear this bit
[7:4]			Reserved.	
3	IISB	R	Interrupt IN If STALL happened, the bit is set. The bit is for debugging only.	
2	BOSB	R	Bulk OUT If STALL happened, the bit is set. The bit is for debugging only.	
1	BISB	R	Bulk IN If STALL happened, the bit is set. The bit is for debugging only.	
0	EP0SB	R	Endpoint0 If STALL happened, the bit is set. The bit is for debugging only.	

#### USB Host Register Definition

Bit 0 in [P\_USBH\_Config] must set to 1 when accessing the USB Host functions.

P_USBH_Config			0x881C0000						USB Host Configuration Register							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	SUS	-	ASOF	SOFTTR	-	HOSTEN
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:6]			Reserved	
5	SUS	R/W	Host Suspend Write "1" to this bit will enable host suspend.	0 = HOST SUSPEND is disable. 1 = HOST SUSPEND is enable. USB Transceiver is in SUSPEND mode. (The SUSPEND mode is controlled by software when HOST is enable.)
4			Reserved	
3	ASOF	R/W	Auto Generate SOF This bit is set to generate SOF timer by hardware. If this bit is 0, write 1 to P_USBH_Transfer[0] to generate SOF timer by software.	0= Generate SOF by software 1= Generate SOF by hardware
2	SOFTTR	R/W	SOF Timer Write "1" to enable SOF timer	0= Disable SOF timer 1= Enable SOF timer



1			Reserved	
0	HOSTEN	R/W	Host Enable Write "1" to this bit to enable host.	0= Host is disabled 1= Host is enabled (Device is disabled)

P_USBH_TimeConfig			0x881C0004										USB Host Timing Configuration Register			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	SAU	PAC	TC		IPD	
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:6]			Reserved	
5	SAU	R/W	Storage 1/2 Auto Mode If this bit is set 1, the following behavior is not necessary: Setting up P_USBH_StorageRST after each transaction in non-DMA mode.	Always set to 1
4	PAC	R/W	Pointer Auto Clear If the bit is set 1, READ/WRITE pointer is automatically reset to 0 after any transaction. However, this mode is automatically disable in DMA mode. The DMA mode is entered by configuring P_USBH_AutoTrans[0] or P_USBH_AutoTrans[1].	Always set to 1
[3:2]	TC	R/W	TimeOut Criteria	00= 16T 01= 18T 10= 20T 11= 22T
[1:0]	IPD	R/W	Inter Packet Delay	00= 2T 01= 4T 10= 6T 11= 8T

P_USBH_Data			0x881C0008										USB Host Data Register			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	HDATA							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	HDATA	R/W	Host Data	

P_USBH_Transfer	0x881C000C	USB Host Transfer Register
-----------------	------------	----------------------------



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	RST	OD1	OD0	ID1	ID0	Setup	SOF
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:7]			Reserved	
6	RST	R/W	Reset Signal RESET signaling, the RESET signaling is not terminated by hardware automatically, CPU must explicitly write 0 to this bit to stop signaling RESET.	1= Generate reset signal 0= Stop reset signal
5	OD1	R/W	Out Data1 Transfer Write 1 to generate OUT DATA1 transfer. This bit is cleared automatically if the transfer is completed.	
4	OD0	R/W	Out Data0 Transfer Write 1 to generate OUT DATA0 transfer. This bit is cleared automatically if the transfer is completed.	
3	ID1	R/W	In Data1 Transfer Write 1 to generate IN DATA1 transfer. This bit is cleared automatically if the transfer is completed.	
2	ID0	R/W	In Data0 Transfer Write 1 to generate IN DATA0 transfer. This bit is cleared automatically if the transfer is completed.	
1	SETUP	R/W	Setup Transfer Write 1 to generate SETUP transfer. This bit is cleared automatically if the transfer is completed.	
0	SOF	R/W	SOF Transfer Write 1 to generate SOF transfer. (SOF is generated by software) This bit is cleared automatically if the transfer is completed.	

P_USBH_DveAddr 0x881C0010										USB Device Address Register						
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	DAddr						
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:7]			Reserved	
[6:0]	Daddr	R/W	Device Address Write the device address to these bits that host want to access.	



**P\_USBH\_DveEP**

**0x881C0014**

**USB Device Endpoint Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	DEP			
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:4]			Reserved	
[3:0]	DEP	R/W	Device Endpoint Write the device endpoint to these bits that host want to access.	

**P\_USBH\_TXCount**

**0x881C0018**

**USB Host Transmit Count Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	TXC						
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:7]			Reserved	
[6:0]	TXC	R/W	Host Transmitting Count Write the number of data to these bits that host wants to transmit.	

**P\_USBH\_RXCount**

**0x881C001C**

**USB Host Receive Count Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	RXC						
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:7]			Reserved	
[6:0]	RXC	R/W	Host Receiving Count The number of data that host received is stored in these bits.	

**P\_USBH\_FIFOInPointer**

**0x881C0020**

**USB Host FIFO Input Pointer Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	HFIP							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0





Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	HFIP	R/W	Host FIFO Input Pointer	

**P\_USBH\_FIFOPutPointer 0x881C0024**

**USB Host FIFO Output Pointer Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	HFOP							
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:8]			Reserved	
[7:0]	HFOP	R/W	Host FIFO Output Pointer	

**P\_USBH\_AutoInByteCount 0x881C0028**

**USB Host Automatic In Transaction Byte Count Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	HAIBC															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:0]	HAIBC	R/W	Host Automatic IN Transaction Byte Count Write the number of IN transaction that had to be generated in these bits. For example, if the host is going to receive 512 bytes from the device, it has to generate $512/64=8$ IN transaction, and write 8 to this register. This register is used only in DMA mode.	When using DMA in USB host, HAIBC must be multiple of 64 bytes.

**P\_USBH\_AutoOutByteCount 0x881C002C**

**USB Host Automatic Out Transaction Byte Count Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	HAOBC															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
-----	----------	------	-------------	-----------



[15:0]	HAOBC	R/W	Host Automatic OUT Transaction Byte Count Write the number of OUT transaction in these bits that had to be generated. For example, if the host is going to transmit 512 bytes to the device, it has to generate $512/64=8$ OUT transaction, and write 8 to this register. This register is used only in DMA mode.	When using DMA in USB host, HAOBC must be multiple of 64 bytes.
--------	-------	-----	--	---

P_USBH_AutoTrans				0x881C0030				USB Host Auto Transfer Register								
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	CAO	CAI	AOX	AIX
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:4]			Reserved	
3	CAO	W	Clear Auto Out Transfer Write "1" to this bit to clear P_USBH_AutoTrans[1]	
2	CAI	W	Clear Auto IN Transfer Write "1" to this bit to clear P_USBH_AutoTrans[0]	
1	AOX	R/W	Auto Out Transfer This bit is set 1 by entering DMA mode for Bulk out. When transaction is surely finished, this bit must be cleared. Note: After DMA is finished (DMA counter reaches 0), the transaction may not be finished yet. Enable this bit, programming the UHAOBCR and then trigger the OUT transfer will start the OUT transaction. This bit is cleared by software.	
0	AIX	R/W	Auto In Transfer This bit is set 1 by entering DMA mode for Bulk in. When transaction is surely finished, this bit must be cleared. Enable this bit, programming the UHAIBCR and then trigger the IN transfer will start the IN transaction. This bit is cleared by software.	

P_USBH_Status		0x881C0034							USB Host Status Register							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	TO	CRC	DE	BS	UP	SH	NH	AH
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
-----	----------	------	-------------	-----------



[15:8]			Reserved	
7	TO	R/W	TimeOut, No Response This flag is set when timeout or no response from the device	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
6	CRC	R/W	CRC16 Error Packet Received This flag is set when In data packet is received with CRC16 error.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
5	DE	R/W	Data Sequence Error Packet Received This flag is set when In data packet is received with data sequence error.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
4	BS	R/W	Bit Stuffing Error Packet Received This flag is set when In data contains bit stuffing error.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
3	UP	R/W	Unkonwn PID Packet Received This flag is set when receiving a packet with unknown PID.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
2	SH	R/W	Stall Handshake Received This flag is set when receiving a stall handshake.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
1	NH	R/W	NACK Handshake Received This flag is set when receiving a NACK handshake.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
0	AH	R/W	ACK Handshake Received This flag is set when receiving an ACK handshake.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag

**P\_USBH\_INT**

**0x881C0038**

**USB Host Interrupt Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	DPO	TRST	TSOFI	ITOK	TXO	VSC	AOX	AIX	RX	TX	SOF	DSC
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:12]			Reserved	
11	DPO	R/W	Device Plug Out Interrupt This interrupt is asserted whenever host has detected device is plug_out. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
10	TRST	R/W	Transmit USB Reset Interrupt This interrupt is asserted whenever host has sent USB RESET signal. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag



9	TSOFI	R/W	Transmit SOF Interrupt This interrupt is asserted whenever host has sent a SOF. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
8	ITOK	R/W	IN Token Transmit Interrupt This interrupt is asserted whenever host has sent an IN Token. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
7	TXO	R/W	Transmit Data Interrupt This interrupt is asserted whenever host has sent a DATA packet. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
6	VSC	R/W	VBUS Status Change Interrupt This interrupt is asserted whenever VBUS status has changed.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
5	AOX	R/W	Automatic Out Transfer Interrupt This interrupt is asserted when the host has transmitted the data out to the device and host has received an ACK from the device.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
4	AIX	R/W	Automatic In Transfer Interrupt	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
3	RX	R/W	Receive Interrupt This interrupt is asserted whenever the host receives a packet form the device. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
2	TX	R/W	Transmit Interrupt This interrupt is asserted whenever the TX task is completed. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
1	SOF	R/W	SOF Interrupt This interrupt periodically generated for every 1ms frame time. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag
0	DSC	R/W	DP DM Status Change Interrupt D+/D- status change interrupt. This interrupt is used to detect the device connection when the host controller is in the idle state. Once the device is plug-in, this interrupt must be disabled. Write 1 to clear the interrupt.	Read 0= Not occurred Read 1= Occurred Write 0= No effect Write 1= Clear the flag

P_USBH_INTEN				0x881C003C				USB Host Interrupt Enable Register								
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	DPO	TRST	TSOFI	ITOK	TXO	VSC	AOX	AIX	RX	TX	SOF	DSC
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
-----	----------	------	-------------	-----------



[15:12]			Reserved	
11	DPO	R/W	Device Plug Out Interrupt Enable	0= Disable 1= Enable
10	TRST	R/W	Transmit USB Reset Interrupt Enable	0= Disable 1= Enable
9	TSOFI	R/W	Transmit SOF Interrupt Enable	0= Disable 1= Enable
8	ITOK	R/W	IN Token Transmit Interrupt Enable	0= Disable 1= Enable
7	TXO	R/W	Transmit Data Interrupt Enable	0= Disable 1= Enable
6	VSC	R/W	VBUS Status Change Interrupt Enable	0= Disable 1= Enable
5	AOX	R/W	Automatic Out Transfer Interrupt Enable	0= Disable 1= Enable
4	AIX	R/W	Automatic In Transfer Interrupt Enable	0= Disable 1= Enable
3	RX	R/W	Receive Interrupt Enable	0= Disable 1= Enable
2	TX	R/W	Transmit Interrupt Enable	0= Disable 1= Enable
1	SOF	R/W	SOF Interrupt Enable	0= Disable 1= Enable
0	DSC	R/W	DP DM Status Change Interrupt Enable	0= Disable 1= Enable

**P\_USBH\_SoftRST      0x881C0044      USB Software Reset Register / Device Plug Out Register**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	DPOE	DPOTV							SRST
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:9]			Reserved	
8	DPOE	R/W	Device Plug Out Timer Enable Write "1" to this bit to enable the timer	
[7:1]	DPOTV	R/W	Device Plug Out Timer Value If DEVICE_PLUG_OUT_TIMER_ENABLE is 1, the inside timer is enable. For each clock cycle, if D+ and D- are all 0, the timer is added by 1. Besides, if one of D+/D- is not 0, the timer is reset to 0. When it counts to DEVICE_PLUG_OUT_TIMER_VALUE, an interrupt is happened and DEVICE_PLUG_OUT_TIMER_ENABLE is reset.	
0	SRST	R/w	Software Reset	



P_USBH_INAckCount			0x881C005C			USB IN ACK Count Register										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	INACK															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:0]	INACK	R/W	IN ACK Count	

P_USBH_OutAckCount			0x881C0060			USB OUT ACK Count Register										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	OUTACK															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:0]	OUTACK	R/W	OUT ACK Count	

P_USBH_RSTAckCount			0x881C0064			USB Reset ACK Count Register										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IARST	OARST
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:2]			Reserved	
1	IARST	W	IN ACK Reset Write "1" to reset P_USBH_INAckCount	
0	OARST	W	OUT ACK Reset Write "1" to reset P_USBH_OUTAckCount	

P_USBH_Dreadback			0x881C006C			USB D+ / D- Readback Register										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DM	DP
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Function	Type	Description	Condition
[15:2]			Reserved	
1	DM	R	VPIN input signal	
0	DP	R	VMIN input signal	

---

## 17 SD CARD CONTROLLER

---

Secure Digital (SD) Memory card is a Flash-Based memory card that is specifically designed to meet the security, capacity, performance and environment requirements inherent in newly emerging audio and video consumer electronic device. The SD Memory Card communication is based on an advanced 9-pin interface (Clock, Command, 4xData and 3xPower lines) designed to operate in a low voltage range.

SD IO card is based on and compatible with the SD memory card. The intent of the SD IO card is to provide high-speed data I/O with low power consumption for mobile electronic devices.

The SD card controller built in SPG290 is designed to have high performance transfer rate using DMA access which can achieve the best performance/cost ratio.

### 17.1 Features

- \* Fully compatible with SD Memory card specification
- \* Accept SD command directly which improve the compatibility.
- \* Programmable clock speed on the SD bus.
- \* SD bus clock control while buffer is full.
- \* Interrupt generation
- \* DMA R/W operation
- \* Both 1-bit, 4-bits SD mode supported
- \* SD IO card interrupt detection.

## 17.2 Block Diagram

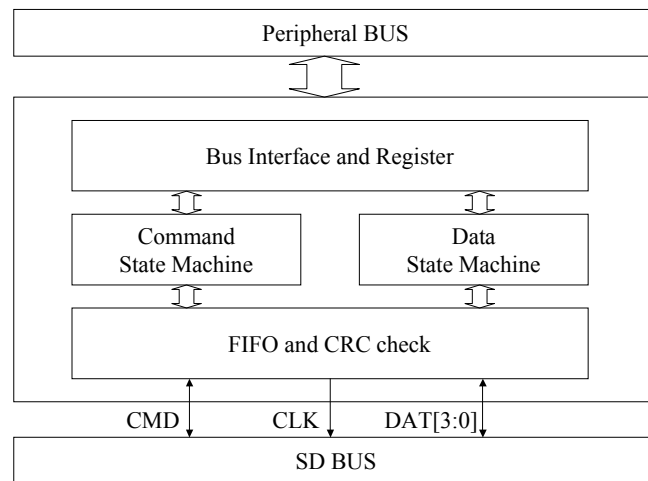


Figure 1. Functional Block Diagram of SD Memory Controller

## 17.3 Control Register

Host use two registers to transfer data to and from SD Card. Host need to poll the DatBufFull /DatBufEmpty bits in the status register to determine if the controller is able to receive/transfer data.

### 17.3.1 P\_SD\_DATATx (R/W)(0x88180000) : SD Card Data Transmit register

Host writes 32-bits data to this register and the controller will transmit it to SD card. When the data stored in the buffer is transmitted, DATBUFEMPTY bit in status register will be set or the DMA request will be issued. It should be noted data could be written to this register only when **DATBUFEMPTY** is 1.

B31-B0
DataTx

DataTx: Write Data to SDCard, data can be written to this register only when DATBUFEMPTY is 1, default is zero.

### 17.3.2 P\_SD\_DATARx(R) (0x88180004): SD Card Data Receive register

This register is used to store the read data from the SD card. When 32-bits data is received, DATBUFFULL bit in status register will be set or the DMA request will be issued. It should be noted data could be read from this register only when **DATBUFFULL** is 1.



B31-B0
DataRx

**DataRx:** Read data from SD card, read data from this register will only valid when the **DATBUFFULL** is set, otherwise, it will return zeros..

### 17.3.3 P\_SD\_COMMAND(R/W)(0x88180008): SD Card Command register

Host uses this register to control the behavior of controller. Controller will use the information in this register to determine what to do.

B7	B6	B5	B4	B3	B2	B1	B0
RUNCMD	STPCMD	CMDCODE					
B15	B14	B13	B12	B11	B10	B9	B8
	RESPTYPE			INICARD	MULBLK	TxData	CMDWD

**CMDCODE:** The Command code host wishes to transfer

**STPCMD:** Write 1 to this bit will force the controller back to IDLE state. This bit will be clear to 0 after te controller back to IDLE state.

**RUNCMD:** Write '1' to this register will initiate the SD command on the SD bus according to current configuration of the controller. This bit will be cleared to '0' after the transaction start. You can start a new transaction only when BUSY bit is 0.

**CMDWD:** ndicate if this command with data.  
0 = Command without data.                      1 = Command with data.

**TRANDATA:** Indicate this command transfer or receive data.  
0 = Receive data (Read).                      1 = Transfer data (Write).

**MUL BLK:** Indicate multi-block transfer or not.  
0 = Single block transfer.                      1 = Multiple block transfer.

**INICARD:** Write This Bit to 1 will start a 74 clock cycles on the clock line.

**RESPTYPE:** Indicate the response type of this command. Currently, only the response type R2 have response length 128 bits, all other response will have 32 bits length. Response type R1b will keep the controller to wait for busy signal on the SD bus.  
000 = No response.                      001 = Response type R1.  
010 = Response type R2.                      011 = Response type R3.



110 = Response type R6.

111 = Response type R1b

#### 17.3.4 P\_SD\_ARGUMENT(R/W)(0x8818000C): SD Card argument register

Host writes the argument it wants to transfer to SD card in this register. Controller will use data in this register as the command argument transfer to the card. The SD Command needs a 32-bits long command.

B31-B0
Argument

Argument: Argument[15:0] transfer to SD Card

#### 17.3.5 P\_SD\_STATUS (R)(0x88180014): SD Card Controller status register

All the controller status and transaction status will be stored in this register. Host can get information of the controller to determine what to do.

B7	B6	B5	B4	B3	B2	B1	B0
DATBUFFULL	CMDBUFFFULL	RSPCRCERR	RSPIDXERR	DATCOM	CMDCOM	CARDBUSY	BUSY
B15	B14	B13	B12	B11	B10	B9	B8
		CARDINT	CARDPRE	CARDWP	DATCRCERR	TIMEOUT	DATBUFEMPTY

BUSY: Indicate the controller is busy.

0 = Controller is idle. 1 = Controller is busy.

CARDBUSY: Indicate the SD card is busy (drive the DAT0 low). Host needs to poll this bit after a write command is issued.

0 = Card is not Busy. 1 = Card is Busy.

CMDCOM: Indicate corresponding response is received or timeout happened after send a command.

DATCOM: Indicate data transfer/receive have completed.

RSPIDXERR: Indicate the command index in the response is failed.

RSPCRCERR: Indicate the CRC bits in the response is failed, this bit will be set if the CRC received is not 6'b111111 in the case of response R3.

CMDBUFFFULL: Indicate the RESP register is full. Read to RESP register or Start a new transaction or set STPCMD in command register will clear this bit.

DATBUFFULL: This bit will be set when data stored in FIFO is higher than the trigger level. This bit will be clear after appropriate number of data been read from the DATARx register or write 1 to STPCMD bit in

	command register.
DATBUFEMPTY:	This bit will be set when data stored in the FIFO is lower than the trigger level. This bit will be clear after appropriate number of data been written to the DATATx register or write 1 to STPCMD bit in command register.
TIMEOUT:	Indicate command response time out or read data response time out.
DATCRCERR:	Indicate read data CRC error or write data with CRC error response.
CARDWP:	Indicate the card is write protect. This bit only detect the write protect pin on the interface. Controller's behavior will not be affected by this bit. Host need to take the responsibility to protect the card.
CARDPRE:	Indicate the card is present. This bit only detect the DAT3 on the SD interface when the controller is idle. Controller's behavior will not be affected by this bit. Host can initial a transaction no matter what this bit is. Write 1 to this register will clear the pending interrupt of card present.
CARDINT:	Indicate a SD IO card interrupt is pending. This bit will be set only when IOEN in control register is 1. Host need to clear the interrupt using device specific command. Write 1 to this register will have no effect.

#### 17.3.6 P\_SD\_RESPONSE (R ) (0x88180010): SD Card Response register

The response of the SD card will be store in this register. Note the controller will not interpret the information in this register. Host driver needs to take care this.

Command with response R1, R1b, R3, R6 have 6 bits command index and 32-bits response length. The Response will be stored in this register. Command with response R2 will have response length 128 bits. Host need to poll the CMDBUFFFULL bit in the status register to determine when to read this register. The data in this register is valid only when CMDBUFFFULL bit is 1.

B31-B0
Response

Response: Response data from SD card, read data from this register will be valid only if the **CMDBUFFFULL** is set.



### 17.3.7 P\_SD\_CONTROL(R/W)(0x88180018): SD Card Control Register

The register control the clock speed on the sd bus, the block length is also control by this register when transfer or receive data. This register is changeable only when BUSY bit in status register is 0.

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
				EN_SD	IOEN	DMAMODE	BUSWIDTH	CLKDIV							
B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16
BLKLEN															

- CLKDIV: The Clock Speed on the SD bus is calculated from this register.  
 $F_{SDCLK} = F_{SYSCLK} / 2(CLKDIV + 1)$ , Default value is 0x54
- BUSWIDTH: Indicate the data bus width during transfer.  
 0 = 1 bit. 1 = 4 bits.
- DMAMODE: Indicate DMA mode or not
- IOEN: 0 = Disable SD IO card interrupt detection.  
 1 = Enable SD IO card interrupt detection.
- EN\_SD: Indicate if the IO is used by the SD controller. Programmer must write this bit to 1 before start using the SD controller.  
 0x0: SDCLK, SDCMD, SDDAT0 is used as GPIO.  
 0x1: SDCLK, SDCMD, SDDAT0 is used as SD control signal.  
 If BUS\_WIDTH is set to 1, the SDDAT1, SDDAT2, SDDAT3 will become SD control signal when this bit is set to 1
- BLKLEN: Data block length to be transferred, in the unit of bytes. The value in this register should be equal to the block length of the SD/MMC card.

### 17.3.8 13.10 P\_SD\_INTEN(R/W) (0x8818001C): SD Card Interrupt Register

This register is used to enable/disable the interrupt of the SD memory card controller.

B15-B7	B6	B5	B4	B3	B2	B1	B0
	INTEN6	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0

- INTEN0: 0:Disable CMDCOM interrupt  
 1:Enable CMDCOM interrupt, fire interrupt when **CMDCOM** set.



INTEN1:	0:Disable <b>DATCOM</b> interrupt. 1:Enable <b>DATCOM</b> interrupt, fire interrupt when <b>DATCOM</b> set.
INTEN2:	0:Disable <b>CMDBUFFFULL</b> interrupt. 1:Enable <b>CMDBUFFFULL</b> interrupt, fire interrupt when <b>CMDBUFFFULL</b> set.
INTEN3:	0:Disable <b>DATBUFFFULL</b> interrupt. 1:Enable <b>DATBUFFFULL</b> interrupt, fire interrupt when <b>DATBUFFFULL</b> set.
INTEN4:	0:Disable <b>DATBUFEMPTY</b> interrupt. 1:Enable <b>DATBUFEMPTY</b> interrupt, fire interrupt when <b>DATBUFEMPTY</b> set.
INTEN5:	0:Disable card insert/remove interrupt. 1:Enable card insert/remove interrupt.
INTEN6:	0:Disable SD IO card interrupt. 1:Enable SD IO card interrupt..

## 18 UNIVERSAL FILE SYSTEM – UFAT LIBRARY

UFAT, Universal FAT, file system can be used in most of the embedded applications with file-like storage. The mainly UFAT API functions of SPG290 UFAT LIB are listed as following

API name	Description
Open	Open the specified file with the specified operation mode.
Read	Read bytes from the file specified by a file node index
Write	Write bytes into the file specified by a file node index
Close	Close the file specified by the file node index

### 18.1 Structure

UFAT is designed as the following structure:

<b>Interface Layer</b>	APIs for the application developers
<b>API Layer</b>	Operations for files and directories
<b>File System Layer</b>	Operation for FAT and clusters
<b>Logic Block Layer</b>	Sector based access and caching
<b>Driver Layer</b>	Low level device access

**Interface Layer:** A thin layer translates the API layer function calls and error numbers into POSIX function calls and error numbers.

**File System Layer:** Operates the FAT and clusters, follow the FAT storage rules.

**Logic Block Layer:** A sector base access interface is defined in this layer to read sectors or write sectors with cache.

**Driver Layer:** Provide a hardware independent access interface for the upper layer. SPG290 UFAT library support both **FAT16** and **FAT32** storage formats

## 18.2 API Functions

### 18.2.1 open

Open a specified file.

int open (const char \*filename, int flags)

#### Parameters:

**Filename:** pointer to a path name string.

#### Flags:

Flags	Description
O_OPEN	Open a exists file
O_TRUNC	Open the file and truncate the file to zero length
O_CREAT	If set, the file will be created if it doesn't already exist
O_RDONLY	Open the file for read access
O_WRONLY	Open the file for write access
O_RDWR	Open the file for both reading and writing
O_EXCL	If set, the open operation will fail when trying to create and existent file

#### Returns value:

When success, the return value of open() is a non-negative integer file handle.

Return -1 when error occurs.

### 18.2.2 close

Close a file handle..

int close (int filehd)

#### Parameters:

**filehd:** integer file handle.

#### Return value:

When success, return 0. -1 when error occurs.



### 18.2.3 read

Read bytes from the specified file handle.

Int read (int filehd, void \*buffer, unsigned int size)

**Parameters:**

**filehd:** Integer file handle returned from open function.  
**buffer:** Pointer to the buffer for storing the data.  
**size:** bytes to read.

**Return values:**

When success, it will return the actual number of bytes readed from the specified file handle. Return -1 when error occurs.

### 18.2.4 write

Write bytes into a specified file handle.

Int write ( int filehd, void \*buffer, unsigned int size)

**Parameters:**

**filehd:** Integer file handle returned from open function.  
**buffer:** Pointer to the buffer for writing.  
**size:** bytes to write.

**Return value:**

When success, it will return the actual number of bytes written to the specified file handle.  
Return -1 when error occurs.

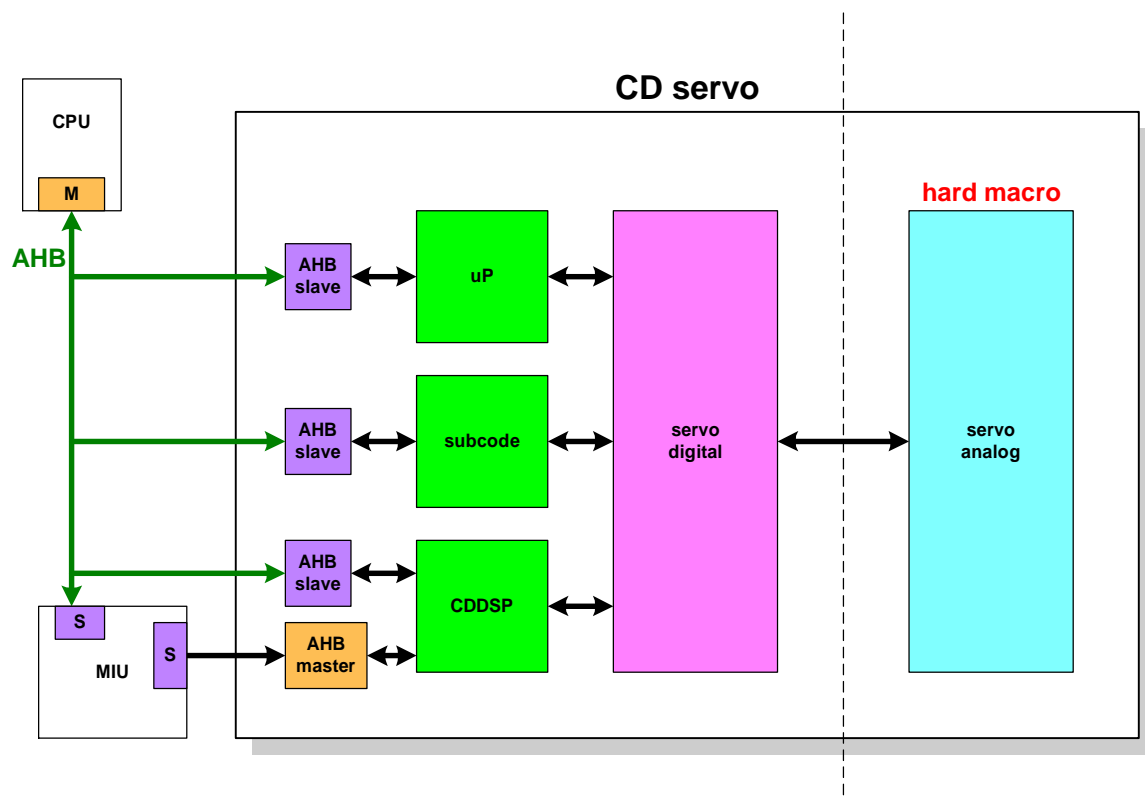


## 19 CD SERVO CONTROL

### 19.1 Architecture

CD module 是取自產發二處的 IP，聯絡人是黃界堅。下圖三個綠色 blocks 及 digital servo, analog servo 是取自 PU9981，其他則是我們自己要加入的 blocks。三個 AHB slave 是要讓 CPU 讀寫位於 uP, subcode, 及 CDDSP 內的 control registers。AHB master 是要讓 CDDSP 將從光碟片上讀到的資料透過 MIU 寫到 external DRAM 裡。

### Block Diagram of CD Servo



### 19.2 Data buffer in DRAM

CD 輸出的是 buffer 內的 local address。MIU 有一個 CD buffer start address，其值由 CPU 從 AHB slave 0 填入，MIU 會把 CD 送出的 local address 加上 CD buffer start address 當作真正的 DRAM address。CD 從 local address 0 往 DRAM 寫資料，當每寫滿 1kB(大小應做成可以調整)就發 interrupt 通知 CPU。

這個 buffer 暫定為 2kB(前述 1kB 的兩倍)，當寫到底會再折回從頭開始寫。我們假定 CPU 收到 interrupt 後把上述 1kB data 讀走的速度很快，所以 CD 就一直往這個 buffer 寫 data，不必擔心 buffer overflow。至於 data 若是不是 1kB 的倍數，最後剩下的 data 該如何通知 CPU 準確地讀走，不會讀到多餘的 data，則仿照 DVD/VCD team 做法，以一個 register 去反應現在填了多少資料在 DRAM 中，CPU 由此 register 去判斷。

### 19.3 Bus interface

CD 有一個 AHB master 及三個 AHB slave(整合成一個比較好吧?)，目的如前所述。為了以後這個 IP 能給其他不同的計畫使用，AHB master 和 slave 都必須做成 fully compliant with AMBA 2.0。

對 SPG290 而言，這個 CD AHB master 只接到 MIU (memory interface unit) 上專屬的 AHB slave，不需要 Arbiter，所以 HGRANT input 會固定接 1，而 HBUSREQ output 會 float。CD AHB slave 不做 RETRY/SPLIT (沒必要)，所以 HMASTER[3:0]，HMASTLOCK，HSPLITx[15:0]這些 port 都不必做。CD AHB master 每次 request 都是 4-beat incrementing burst，一次 burst access 16 bytes，也就是 HBURST 只會是 HBURST[2:0] = 3'b011 (4-beat incrementing burst)

CD AHB master 的 HSIZE[2:0] 一定是 3'b010 (32 bits)，CD AHB slave 的 HSIZE[2:0]有 3'b000 (8 bits)，3'b001 (16 bits) 和 3'b010 (32 bits) 三種可能。

AHB clock frequency 與 CD main clock frequency 沒有倍數關係，且不假設誰快誰慢。

### 19.4 Control registers

CD 中的 control registers 應包括 (但不限於) Data length (告訴 CPU 有多少資料存放在 DRAM 裡的数据 buffer)

CD 開始要與其他 module 整合時，負責人員要提供完整的 register map 及 function description。

### 19.5 Interrupt

Interrupt register 是由 AHB clock 驅動，當某些 event 發生時，硬體會將此 register 變成 1。

CD 的 interrupt 有 mask register 可設定，當此 register 填 1 時，interrupt 不會傳出 CD。

CPU 收到 interrupt 會來讀 status register，硬體會利用讀 status register 的 event 來清除 interrupt。



## 20 LCD TIMING CONTROL

### 20.1 Frame buffer

Frame buffer 一定是位於 external DRAM 裡，因為 embedded SRAM 容量不足以儲存整張 frame。Frame buffer 內的資料是採 raster-scan 順序存放。LCD 只輸出 frame buffer 內的 local address。MIU 內有兩個 LCD frame buffer start address register 作為 double buffer 之用，address 由 CPU 從 AHB slave 0 填入。MIU 會把 PPU 送出的 local address 加上 frame buffer start address，至於 PPU 要寫到哪一個 frame buffer 是由 BUFCTL module 控制。

一條 scan line 有幾個 pixel 記錄於 control register 中。存放在 frame buffer 裡的 image data 有兩種格式：YUYV 和 RGB565。LCD 每次從 AHB master 讀到的 image data 一定是 32-bit word，一個 32-bit word 內含 YUYV 或是 RGB565，代表 2 個 pixel 的資料。

### 20.2 Buffer control

LCD 從 DRAM 讀出的 frame 是由 PPU、CSI、JPG 寫出，這些 module 的 frame rate 未必與 LCD 相同或是有倍數關係，所以 LCD 與這些 module 之間必須做 buffer control，以避免 data loss。所有的 buffer control 都交由 BUFCTL 這個 module 負責。對 LCD 而言，只要有以下的 output port 即可：

name	I/O	function description
LCD_FRAME_END	Output	A pulse to indicate that LCD has completed the read transfer of a frame.

LCD\_FRAME\_END 是 AHB master clock 驅動。

BUFCTL module 有以下的 I/O port：

name	I/O	function description
LCD_FRAME_END	Input	A pulse to indicate that LCD has completed the read transfer of a frame.
LCD_BUFFER_PTR	Output	0: read LCD frame buffer A 1: read LCD frame buffer B

上表的 LCD\_BUFFER\_PTR 接到 MIU。

### 20.3 Bus interface

LCD 有一個 AHB master 及一個 AHB slave，前者目的是要讓 LCD read frame buffer（位於 external DRAM），後者是要讓 CPU 讀寫 LCD 中的 control registers。為了以後這個 IP 能給其他不同的計畫使用，AHB master 和 slave 都必須做成 fully compliant with AMBA 2.0。

對 SPG290 而言，這個 LCD AHB master 只接到 MIU（memory interface unit）上專屬的 AHB slave，不需要 Arbiter，所以 HGRANT input 會固定接 1，而 HBUSREQ output 會 float。LCD AHB slave 不做 RETRY/SPLIT（沒必要），所以 HMASTER[3:0]，HMASTLOCK，HSPLITx[15:0]這些 port 都不必做。

LCD AHB master 每次 request 一定是 8-beat incrementing burst，也就是 HBURST 只會是

HBURST[2:0] = 3'b101 (8-beat incrementing burst)

LCD AHB master 的 HSIZE[2:0] 一定是 3'b010 (32 bits)，LCD AHB slave 的 HSIZE[2:0] 有 3'b000 (8 bits)，3'b001 (16 bits) 和 3'b010 (32 bits) 三種可能。

AHB master clock frequency 與 LCD main clock frequency 沒有關係，但一定高於或等於 27MHz。

AHB slave clock frequency 與 LCD main clock frequency 沒有關係，但一定高於或等於 27MHz。

所以 AHB master clock domain 與 TFT/CSTN main clock domain 之間要做同步電路。AHB slave clock domain 與 TFT/CSTN main clock domain 之間也要做同步電路。

## 20.4 Clock input ports

LCD 有四個 clock input ports。

AHB master clock

AHB slave clock

TFT main clock (20MHz ~ 27MHz)

CSTN main clock (54MHz)

## 20.5 Control registers

LCD 中的 control registers 係由 CPU 從 AHB slave 寫入/讀出。

Scan line length (DRAM 中一條 scan line 空間可存放幾個 pixels)

LCD status

VGA mode (0 表示解析度是 320x240，1 表示是 640x480。Reset 值為 0)

Data format (0 表示存放於 frame buffer 的資料格式是 RGB，1 表示是 YUYV)

LCD 必須有一個 disable reg，當 CPU 填 1 時，LCD 會等目前對 MIU 的 AHB transfer 完成後停止 AHB transfer。停止 AHB transfer 後 LCD 會以 status register 的一個 bit 讓 CPU 知道已停止對 MIU 的 AHB transfer。

目前我只列了我想到的，當 LCD 開始要與其他 module 整合時，請提供完整的 register map 及 function description。

## 20.6 Interrupt

Interrupt register 是由 AHB clock 驅動，當某些 event 發生時，硬體會將此 register 變成 1。

LCD 的 interrupt 有 mask register 可設定，當此 register 填 1 時，interrupt 不會傳出 LCD。

CPU 收到 interrupt 會來讀 status register，硬體會利用讀 status register 的 event 來清除 interrupt。

## 20.7 Supported LCD panels

LCD 支援 TFT 以及 CSTN LCD panels，支援的面板種類比照 SPG220。



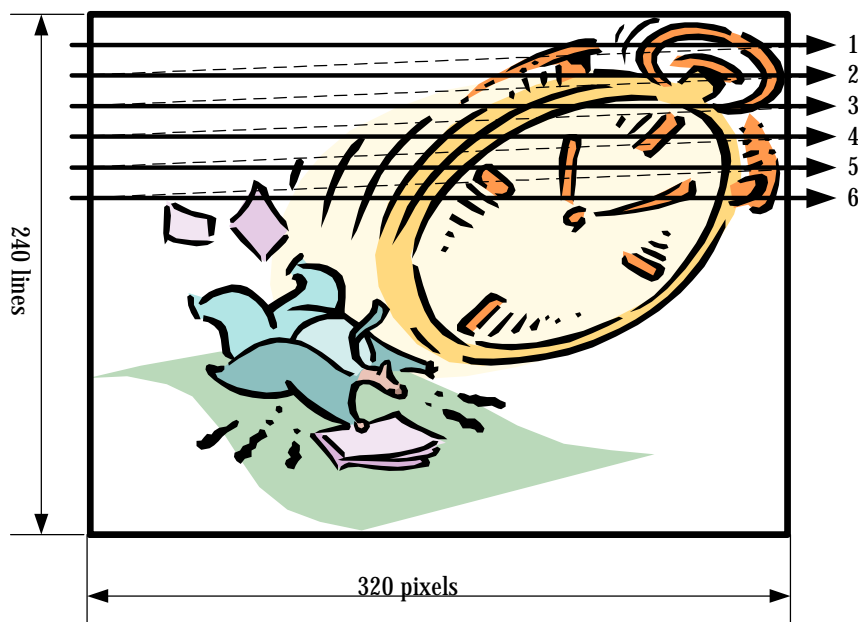
## 21 LIGHT GUN APPLICATION

### 21.1 Introduction

The X,Y position of the lightgun can be read through IO port without other circuit. Principle of lightgun and the sample program can be found in the following description.

There are 3 scan modes supported for TV: interlace, non-interlace, and progressive modes.

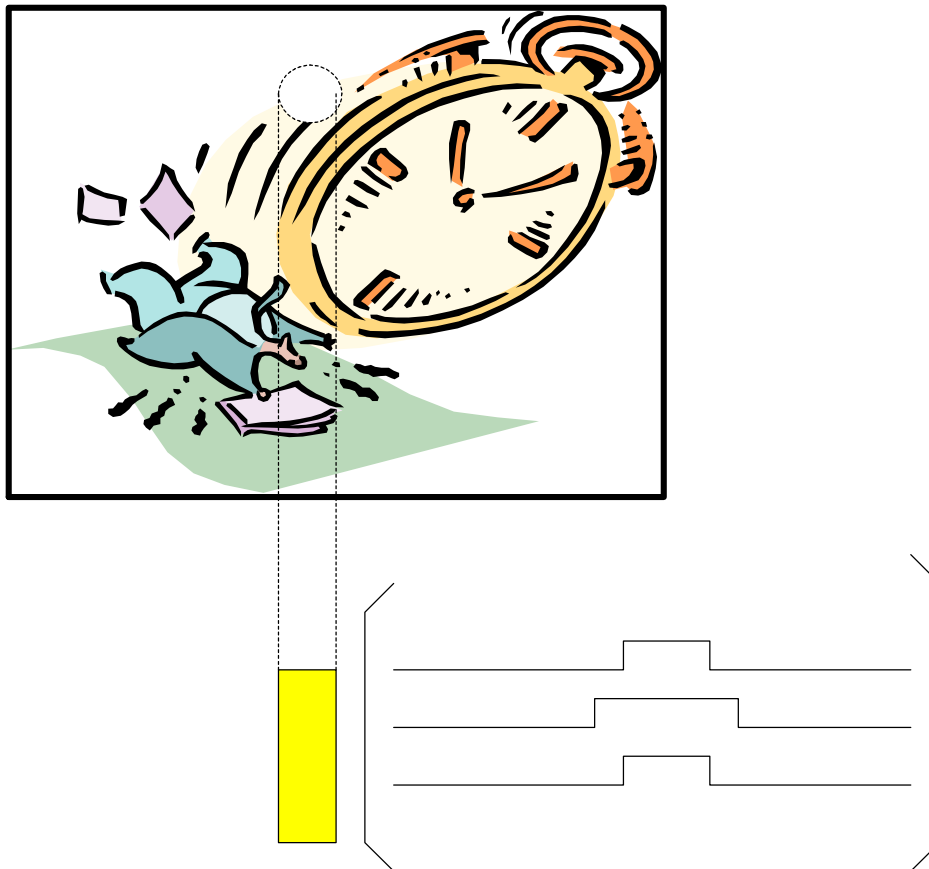
- a. Interlace: Scan all the odd lines, then scan all the even lines.
- b. Non-interlace: Scan the odd/even lines only.
- c. Progressive: Scan each line from top to bottom.





## 21.2 Principle of Lightgun

Recur to the different scanning mode on TV, the corresponding position of lightgun is caught by SPG290 as the following figure. Three high-pulse areas indicate the lightgun position is caught by the scan line.



## 21.3 Test Environment

1. Gun and TV distance: 1M
2. Gun Target Sprite Size : 24 \* 24
3. Enemy Sprite Size :
  - a. 64 \* 64 16 color
  - b. 96 \* 96 16 color
  - c. 128 \* 128 16 color
4. BG1 & BG2 mode and size: 512 \* 256 64 color Character Mode

## 21.4 Lightgun Control

Control method:

SPG290 : 0x283E, 0x283F

Ordinate: 0x283E / 4      Abscissa: 0x283F / 4

## 21.5 Lightgun Function

Lightgun function provides x, y positions of lightgun and LightGun Shot. Must write the x, y positions to R\_LPH\_buf[ ] and R\_LPV\_buf[ ] from V-blanking. GetGunAxis() can read the x, y positions. Because the Port 0x283E and 0x283F are sensitive, GetGunAxis() reads the R\_LPH\_buf[ ] and R\_LPV\_buf[ ] average coordinate value of 4 times of gunshot to be the real x, y positions, and the relevant variables are Gun\_X and Gun\_Y. We used the ReadKey() to detect the key input of the gun shot key. The key must be connected to the corresponding IO port through circuit. The key input operation triggers on the input. Suppose IOB7 is set as the input of gunshot.

**void IRQ52(void)**

```
{  
    UINT32 status;  
    status = *P_TV_IRQ_STATUS;  
    *P_TV_IRQ_STATUS = status;  
    lightgun_time++;  
    print1("IRQ 52: TVE Light Gun!\n");  
}
```

**\_GetGunAxis:**

// Set Guns Address Key

## 21.6 Example Program

Example posited in

Suppose the text is City and the sprite is Enemy. The input device is lightgun and IOB7 is the gunshot input from lightgun.

Circuit connection:

- LightGun sensor is connected with IOA15 and IRD1.
- LightGun SHOT is connected with IOB7.
- One wire is connected to VCC and one wire is connected to GND.



Step 1: Irq0 will be read the x, y position and detect gunshot of lightgun, then write the x coordinate to the R\_LPH\_buf,

write the y Coordinate to the R\_LPV\_buf.

Step 2: GetGunAxis() will be average the R\_LPH\_buf to get the real x coordinate value of lightgun and average the

R\_LPV\_buf to get the real y coordinate value of lightgun.

Step 3: ChangeGunAxis() will reset the cursor coordinate on the screen by the real x, y coordinate of lightgun.

Step 4: When Press the gunshot input, ControlKey() will set the HitEnemySw=1, then will run CheckEnemy() to detect the

cursor coordinate touch or no touch the Enemy rectangle. If touch anyone, that enemy will run die action.

This Example the Enemy have 3 type Sprite: a. 128\*128

b. 096\*096

c. 064\*064

```
void CheckEnemy(void)
{
    int x,y,it;
    if(HitEnemySw==1)           // when press the gun shot key then the HitEnemySw=1
    {
        x=GunX+Screen_X+16;    // fix the x coordinate
        y=GunY+Screen_Y+16;    // fix the y coordinate
        for(i=14; i<17; i++)
        {
            if(Sprite_Buff[i].SpType==0)    // check the SpType
                continue;
            switch(Sprite_Buff[i].SpType)
            {
                case 1:    // 128
                    Hit_Lib(i, 64, x, y);    // check hit the 1-level enemy?
                    break;
                case 2: // 096
                    Hit_Lib(i, 48, x, y);    // check hit the 2-level Enemy?
                    break;
                case 3: // 064
```





```
        Hit_Lib(i, 32, x, y);    // check hit the 3-level Enemy?
        break;

    }

    if(TargetSw==1)                // if hit anyone of enemy rectange then break
        break;

}

}

HitEnemySw=0;
}
```

## 22 BUFFER CONTROL - BUFCTL

Buffer control unit is a comprehensive hardware logic to coordinate all double or triple buffers switch in TV encode unit and PPU TEXTs. When PPU output double buffers with hardware buffer control function is enabled, buffer control unit will change the display area pointer in the TV encoder to the other PPU output buffer whenever it is fully repainted. Therefore, TV encoder will always project the updated PPU output frame buffer to TV screen automatically without any CPU program required for buffer switch, this will simplify the program flow and reduce the possible timing or interrupt function overhead problems in the system.

### 22.1 Control Registers

#### 22.1.1 P\_PTR\_SETTING(R/W)(0x88090004): Control type

B7	B6	B5	B4	B3	B2	B1	B0
TVE_Ctrl			PPU_Ctrl	Text3_Ctrl	Text2_Ctrl	Text1_Ctrl	

Text1\_Ctrl: Software/Hardware buffer control select

0: use software buffer control for PPU TEXT1

1: use hardware buffer control for PPU TEXT1

Text2\_Ctrl: Software/Hardware buffer control select

0: use software buffer control for PPU TEXT2

1: use hardware buffer control for PPU TEXT2

Text3\_Ctrl: Software/Hardware buffer control select

0: use software buffer control for PPU TEXT3

1: use hardware buffer control for PPU TEXT3

PPU\_Ctrl: Software/Hardware buffer control select

0: use software buffer control for PPU

1: use P2T (PPU to TV encoder) hardware buffer control for PPU double buffer switch

TVE\_Ctrl: Software/Hardware buffer control select

00: use software buffer control for TVE

10: use P2T (PPU to TV encoder) hardware buffer control for TVE

\* P2T mode means TV encoder will access one of the PPU output buffer which has just been updated by PPU for displaying to TV screen automatically



#### 22.1.2 P\_PPU1\_Buf\_PTR\_REG(R/W)(0x8809000C): Software buffer point for Text1

B7	B6	B5	B4	B3	B2	B1	B0
						TEXT1_PTR	

TEXT1\_PTR: Write this register to set which is the active frame buffer from 0 to 2 of PPU

TEXT1;

Read this register to get which is the active frame buffer from 0 to 2 of PPU TEXT1

0: TEXT1 frame buffer0

1: TEXT1 frame buffer1

2: TEXT1 frame buffer2

#### 22.1.3 P\_PPU2\_Buf\_PTR\_REG(R/W)(0x88090010): Software buffer point for Text2

B7	B6	B5	B4	B3	B2	B1	B0
						TEXT2_PTR	

TEXT2\_PTR: Write this register to set which is the active frame buffer from 0 to 2 of PPU

TEXT2;

Read this register to get which is the active frame buffer from 0 to 2 of PPU TEXT2

0: TEXT2 frame buffer0

1: TEXT2 frame buffer1

2: TEXT2 frame buffer2

#### 22.1.4 P\_PPU3\_Buf\_PTR\_REG(R/W)(0x88090014): Software buffer point for Text3

B7	B6	B5	B4	B3	B2	B1	B0
						TEXT3_PTR	

TEXT3\_PTR: Write this register to set which is the active frame buffer from 0 to 2 of PPU

TEXT3;

Read this register to get which is the active frame buffer from 0 to 2 of PPU TEXT3

0: TEXT3 frame buffer0

1: TEXT3 frame buffer1

2: TEXT3 frame buffer2

#### 22.1.5 P\_TVE\_Buf\_PTR\_REG(R/W)(0x88090020): Software buffer point for TVE

B7	B6	B5	B4	B3	B2	B1	B0
						TVE_PTR	

TVE\_PTR: Write this register to set which is the active frame buffer from 0 to 2 of TVE;



Read this register to get which is the active frame buffer from 0 to 2 of TVE

- 0: TVE frame buffer0
- 1: TVE frame buffer1
- 2: TVE frame buffer2

#### 22.1.6 P\_P2T\_SETTING(R/W)(0x8809003C): PPU to TVE buffer control register

B7	B6	B5	B4	B3	B2	B1	B0
					T2_HARD	T1_HARD	P2T_EN

P2T\_EN: Enable P2T hardware buffer control

0: Disable

1: Enable

T1\_Hard: Select the module to writes data

0: PPU

T2\_Hard: Select the module to reads data

0: TVE

#### 22.1.7 P\_BUFCTL\_SETTING(R/W)(0x8809004C): Buffer Control Register

B7	B6	B5	B4	B3	B2	B1	B0
							BUFCTL_IRQ

BUFCTL\_IRQ: 1: Enable BUFCTL interrupt

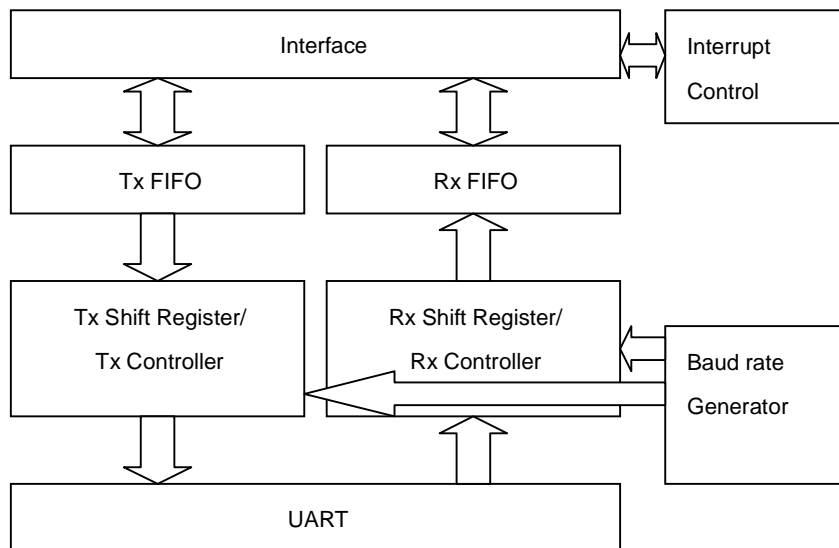
## 23 UART

The UART of SPG290 is an enhanced serial communication unit for more powerful and flexible data exchanges between MCUs. Major features are listed as following:

- \* The maximum baudrate is up to 460.8Kbps.
- \* Programmable of transmitting wait-time and receiving latency-time.
- \* Embedded 2 bytes transmit FIFO and embedded 8 bytes receiving FIFO.
- \* Programmable receiving FIFO size from no FIFO to 8 bytes.
- \* Independent mask of transmit FIFO, receiving FIFO, and receiver timeout interrupts.
- \* False start bit detection.
- \* Link break generation and detection.
- \* Standard MODEM control signals CTS, RTS, DSR, DTR and DCD included.
- \* Fully-programmable serial interface characteristics:
  - Data can be 5, 6, 7 or 8 bits
  - Even , odd or no-parity bit generation and detection
  - 1 or 2 stop bit generation

### 23.1 Block Diagram

Block Diagram of UART Module.



## 23.2 Control Registers

### 23.2.1 P\_UART\_Data(R/W)(0x88150000): UART Data Tx/Rx Register.

B7	B6	B5	B4	B3	B2	B1	B0
UART data							

UART\_Data: UART data Read/Write Register

### 23.2.2 P\_UART\_ERR(R/W)(0x88150004):UART Tx & Rx Error Flag Register

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
RxD												OE	BE	PE	FE

RxD: UART RxD Signal

OE: Overrun Error

This bit is set to 1 if data is received and the FIFO is already full.

Read 0 = Not Occurs

Read 1 = Happened

Write 1 = Clear this error flag

BE: Break Error

This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits).

Read 0 = Not Occurs

Read 1 = Happened

Write 1 = Clear this error flag

PE: Parity Error

This bit is set to 1 if the parity of the received data character does not match the parity selected in PSEL control bit. This bit is refreshing in every read. So, it is necessary to check this bit after DATA register is read.

Read 0 = Not Occurs

Read 1 = Happened

Write 1 = Clear this error flag.

FE: Frame Error

This bit is set to 1 if the received character did not have a valid stop bit (a valid



stop bit is 1 bit). This bit is refreshing in every read. So, it is necessary to check this bit after DATA register is read.

Read 0= Not Occurs

Read 1= Happened

Write 1= Clear this Error Flag

### 23.2.3 P\_UART\_Ctrl(R/W)(0x88150008):UART Control Register

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
RIEN	TIEN	RT	UEN	MSIE					WLSEL	FEN	SBSEL	PSEL	PEN		SB

RIEN: Receive Interrupt Enable

0= Disable 1= Enable

TIEN: Transmit Interrupt Enable

0= Disable 1= Enable

RT: Receive Timeout Interrupt Enable

0= Disable 1= Enable

UEN: UART Enable

0= Disable 1= Enable

WLSEL: Word Length Definition

Indicate number of data bits transmitted or received in a frame

00 = 5bits 01 = 6bits

10 = 7bits 11 = 8bits

FEN: FIFO Buffer Enable/Disable

Enable FIFO buffer during transmit and receive. When clear this bit to "0", the FIFO become 1-byte-deep holding registers

0= Disable 1= Enable

SBSEL: Stop Bit Size Selection

When this bit is set to "1", two stop bits are transmitted at the end of the frame.

The receive logic does not check for two stop bits being received.

0= 1 Stop Bit 1= 2 Stop Bit

PSEL: Parity Selection

If this bit is set to "1", even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to "0" the odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by PEN Control



bit is clear to "0"

0= Odd Parity (if PEN= 1) 1= Even Parity (if PEN= 1)

PEN: Parity Enable

If this bit is set to "1", parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.

0= Disable 1= Enable

SB: Send Break

If this bit is set to "1", a low level is continually output on the TX output pin, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a break condition. The transmit FIFO contents remain unaffected during a break condition. For normal use, this bit must be cleared to "0"

0= Normal Operation 1= Send Break Signal

#### 23.2.4 P\_UART\_BaudRate(R/W)(0x8815000C):UART Baud Rate Setup Register

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
Baud rate															

To get a desired Baud Rate, write the corresponding value to the Port \$8815000CH according to the following Baud Rate equation.

$$\text{Baud Rate} = \text{Fosc} / \text{Baud Rate register} - 1$$

#### 23.2.5 P\_UART\_Status(R/W)(0x88150010):UART Status Register

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
RI	TI	RT	MIT		URI	RTS	DTR	TE	RF	TF	RE	BY	DCD	DSR	CTS

RI: Receive Interrupt Flag

Read 0= Not Occurs Read 1= Happened

TI: Transmit Interrupt Flag

Read 0= Not Occurs Read 1= Happened

RT: Receive Timeout Interrupt Flag

Read 0= Not Occurs Read 1= Happened

MIT: Modem Status Interrupt Flag

Read 0= Not Occurs Read 1= Happened





- URI: This bit is the complement of the uUARTRI modem status input  
0= nUARTRI is 1                      1= nUARTRI is 0
- RTS: Modem output
- DTR: Modem output
- TE: Transmit FIFO Empty Flag  
The meaning of this bit depends on the state of the FEN control bit.  
If the FIFO is disabled, this bit is set to "1" when the transmit holding register is empty  
If the FIFO is enabled, this bit is set to "1" when the transmit FIFO is empty  
0= Not Empty                      1= Empty
- RF: Receive FIFO Full Flag  
The meaning of this bit depends on the state of the FEN control bit.  
If the FIFO is disabled, this bit is set to "1" when the receive holding register is full  
If the FIFO is enabled, this bit is set to "1" when the receive FIFO is full  
0= Not Full                      1= Full
- TF: Transmit FIFO Full Flag  
The meaning of this bit depends on the state of the FEN control bit.  
If the FIFO is disabled, this bit is set to "1" when the transmit holding register is full  
If the FIFO is enabled, this bit is set to "1" when the transmit FIFO is full  
0= Not Full                      1= Full
- RE: Receive FIFO Empty Flag  
The meaning of this bit depends on the state of the FEN control bit.  
If the FIFO is disabled, this bit is set to "1" when the receive holding register is empty  
If the FIFO is enabled, this bit is set to "1" when the receive FIFO is empty  
0= Not Empty                      1= Empty
- BY: BUSY  
If this bit is set to "1", the UART module is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register.  
0= Not Busy                      1= Busy
- DCD: This bit is the complement of the uUARTDCD modem status input  
0= nUARTDCD is 1                      1= nUARTDCD is 0
- DSR: This bit is the complement of the uUARTDSR modem status input  
0= nUARTDSR is 1                      1= nUARTDSR is 0
- CTS: This bit is the complement of the nUARTCTS modem status input.  
0= nUARTCTS is 1                      1= nUARTCTS is 0

## 24 SERIAL INTERFACE I/O - SIO

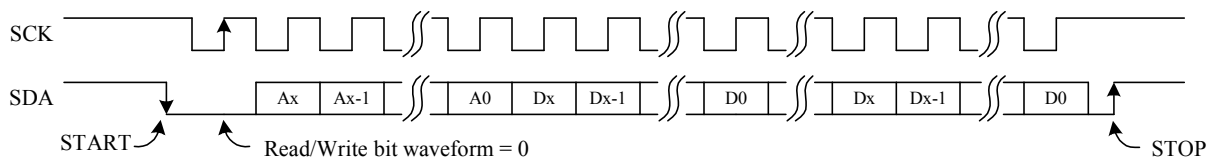
### 24.1 General Description

SIO (Serial Interface I/O) is Sunplus' s proprietary serial interface, which can be used to communicate with other devices. This serial interface is capable of transmitting or receiving data via 2 pins, SCK and SDA.

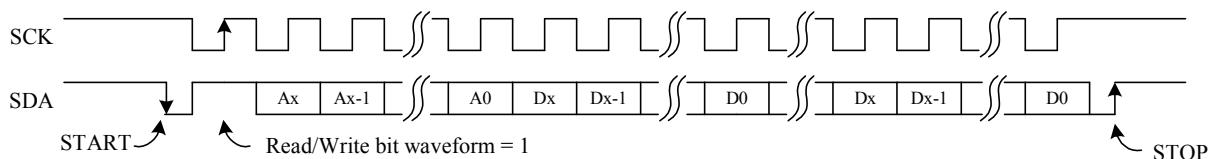
### 24.2 SIO Protocol

The SIO protocol is composed of start bit, read/write control bit, address word, data word, and stop bit. The figures below show the SIO protocol for the read and write mode respectively. If the SDA make a transition from 1 to 0 during the SCK is high, it stands for the “start bit” of the SIO transfer. If the SDA make a transition from 0 to 1 during the SCK is high, it means the “stop bit” occurred. In the SIO waveform other than the “start & stop bit” , the SDA can only change its logic level during the SCK is low. User should note that the address word & data word are transmitted with MSB first. And if the data word is 16 bits in width, then the **low** byte is transmitted/received first.

#### SIO Write Mode :



#### SIO Read Mode :



### 24.3 SIO Feature

The SIO interface support the following features.

1. Support “master read” and “master write” function via SCK and SDA.
2. Support 4 address modes (No address, 8-bit address, 16-bit address, 24-bit address).
3. Support burst reading or writing function.
4. Support 8-bit and 16-bit data width.
5. Support interrupt and polling function.
6. Support 4 SIO baud rates.

7. Provide automatic bit-stream transmitting mode when communicating with SPDS301 IC.

## 24.4 SIO Control Registers

### 24.4.1 P\_SIO\_Control(0x88120000)

NAME	D7	D6-D3		D2	D1	D0
P_SIO_Control	SIO_IRQ_CLR			SIO_TRANSFER	SIO_RW_CTL	SIO_START
NAME	D15	D14	D13	D12	D11-D10	D9-D8
P_SIO_Control	SIO_DATA_WIDTH	APBDMA_EN	IRQ_EN	PROTOCOL	BAUDRATE	ADDR_MODE
NAME	D23-D16					
P_SIO_Control						
NAME	D31	D30	D29-D24			
P_SIO_Control	SIO_REQUEST	SIO_IRQ_STS				

SIO\_START: SIO Start/Stop control bit

0: Stop the SIO transmitting/receiving

1: Start the SIO transmitting/receiving

SIO\_RW\_CTL: SIO Read/Write control bit

0: SIO read mode

1: SIO write mode

SIO\_TRANSFER: Transfer data low byte first

0: SIO transfer Data Register High Portion(0x8812000D) before Data Register Low Portion(0x8812000C)

1: SIO transfer Data Register Low Portion(0x8812000C) before Data Register High Portion(0x8812000D)

SIO\_IRQ\_CLR: SIO Interrupt clear bit

ADDR\_Mode: SIO addressing mode selection bits.

00 = 16-bit address.

01 = No address.

10 = 8-bit address.

11 = 24-bit address.

BAUDRATE: SIO baud rate selection bits.

00 = 27/16 MHz.

01 = 27/4 MHz.

10 = 27/8 MHz.

11 = 27/32 MHz.

PROTOCOL: SIO protocol read/write bit waveform definition.

The definition of the SIO protocol read/write bit is shown in the "SIO Protocol" section.

0 = In write mode, the read/write bit waveform will be low.



In read mode, the read/write bit waveform will be high.

1 = In both write and read mode, the read/write bit waveform will be low.

IRQ\_EN: SIO interrupt enable control bit.

0 = SIO polling method.

1 = SIO interrupt enable. The SIO interrupt use IRQ44.

APBDMA\_EN: APBDMA enable

0 = SIO transfer data by CPU.

1 = Enable SIO transfer data by APBDMA. The detail step is referred to APBDMA.

SIO\_DATA\_WIDTH: SIO data width control bit.

0 = data width is 8 bits.

1 = data width is 16 bits.

SIO\_IRQ\_STS: SIO interrupt status bit.

If this bit is 1, it means the SIO interrupt occurs. This bit is read only.

SIO\_REQUEST: SIO data transfer request bit.

Read only. This bit is useful when you use the polling method instead of the interrupt method. If this bit is read as 1 in the write mode, it means the SIO interface circuit is waiting for the user to write a 8-bit/16-bit data into SIO\_DATA register. If this bit is read as 1 in the read mode, it means the SIO interface circuit has received a 8-bit/16-bit data and store it in the SIO\_DATA register. And user must read the SIO\_DATA register to get the received data.

**Note:** If “automatic bit-stream transmitting mode” is turn on, then this bit is always 0.

#### 24.4.2 P\_SIO\_AutoTx (0x88120004): Automatic transmit word number

NAME	D7-D4	D3-D2	D1	D0
P_SIO_Control	SIO_WORD_NUM		DS301_READY	AUTOMATIC_EN

AUTOMATIC\_EN: Automatic bit-stream transmitting mode for the SPDS301.

0 = disable the automatic bit-stream transmitting mode.

1 = start the automatic bit-stream transmitting mode.

The SIO interface circuit will keep on polling the SPDS301's status register until the “decode work bit” and “request ready bit” are both 1. Upon the completion of the polling, a SIO interrupt will occur, and the user should write a bit-stream data word to the SIO\_DATA register. After writing data to the SIO\_DATA register, this interrupt will be cleared automatically. In summary, each time the SIO interrupt occurs during the automatic bit-stream transmitting mode,

just writes bit-stream data into the SIO\_DATA register. For the detailed information of the SPDS301, please refer to the SPDS301 programming guide.

DS301\_READY: Use DS301\_ready pin.

0 = Not use.

1 = Use.

The DS301\_ready pin may locate at IOB[0] or IOC[10] according to the SIO port select setting. This bit is only effective when

“automatic bit-stream transmitting mode for the SPDS301” is turn on. For proper usage of this control bit, refer to the “SIO usage procedure” section.

SIO\_WORD\_NUM: Automatic Transmit Word Number

Number of data word that will be transmitted to SPDS301 in one package. Default value is 15. This register is only effective when

“automatic bit-stream transmitting mode for the SPDS301” is turn on. The value of this register should **not** be set as 0. Otherwise, it will cause un-predictable result.

#### 24.4.3 P\_SIO\_Addr(0x88120008): SIO Start Address Register

NAME	D23-D16	D15-D8	D7-D0
P_SIO_Addr	SIO_ADDRH	SIO_ADDRM	SIO_ADDRL

SIO\_ADDRL: SIO Start Address Register Low Portion

It is used when addressing mode is not equal 01

SIO\_ADDRM: SIO Start Address Register Middle Portion

It is used when addressing mode is equal 00 and 11

SIO\_ADDRH: SIO Start Address Register High Portion

It is used when addressing mode is equal 11

#### 24.4.4 P\_SIO\_DATA(0x8812000C): SIO Data Register.

NAME	D15-D8	D7-D0
P_SIO_DATA	SIO_DATAH	SIO_DATAH

SIO\_DATAH: SIO Data Register High Portion

SIO\_DATAH: SIO Data Register High Portion

It is used when data width is equal 1.



## 24.5 SIO Interrupt Processing

The SIO interrupt signal is controlled by the 0x8812\_0001[5]. If 0x8812\_0001[5] is 1, the SIO interrupt is enabled. User should note that setting 0x8812\_0000[7] could clear the SIO interrupt signal.

Following tables depict what the proper action is.

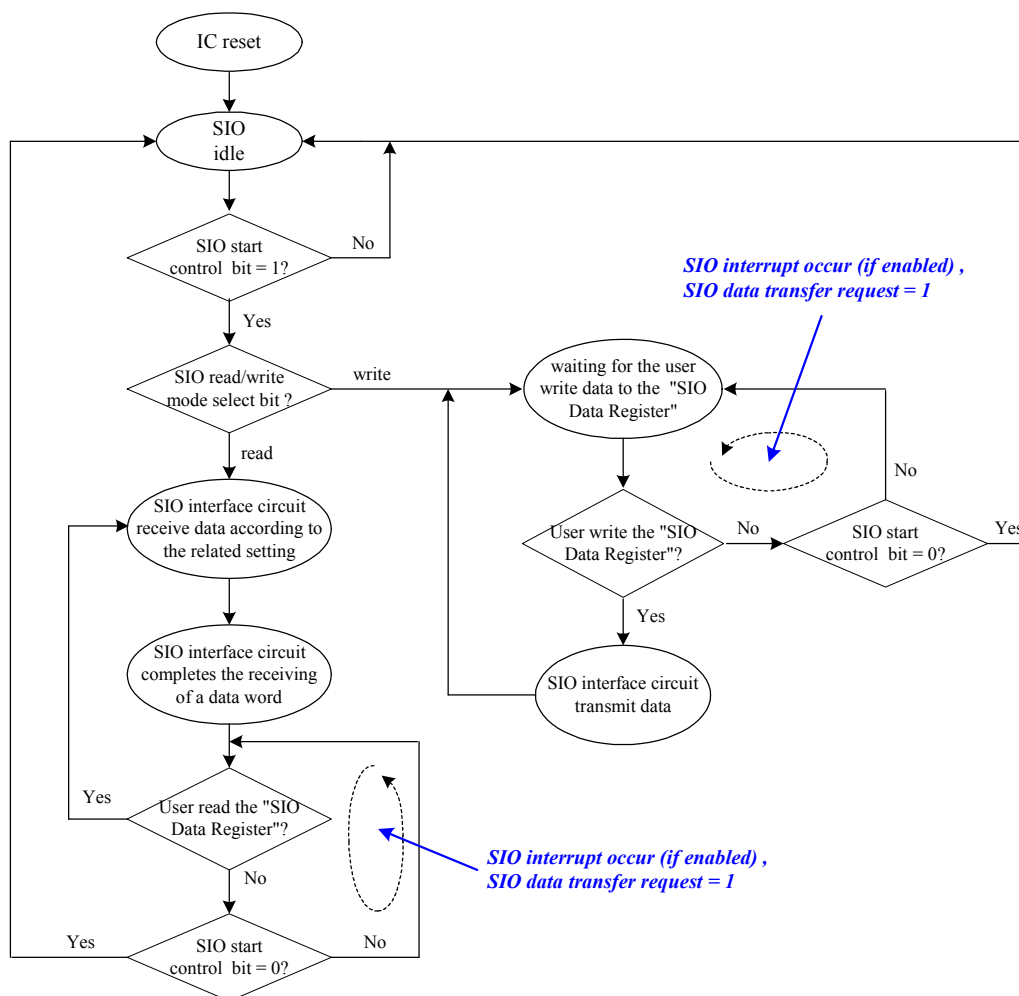
Not using “automatic bit-stream transmitting mode”	
the SIO status when the interrupt occurs	proper action
If SIO interrupt is generated during the “Read Mode”, it means the SIO interface has received a data byte/word.	(a) Reading the SIO_DATA register, and then setting the interrupt clear register clear interrupt. And the SIO interface will continue receiving data. (b) Even clear the start/stop control bit of the SIO, the interrupt is still cleared by setting the interrupt clear register.
If SIO interrupt is generated during the “Write Mode”, it means the SIO interface is waiting for data to be transmitted.	(a) By writing data to the SIO_DATA register, the interrupt will happen after finishing transferring data. And setting the interrupt clear register to clear interrupt. (b) Even clear the start/stop control bit of the SIO, the interrupt is still cleared by setting the interrupt clear register.

Using “automatic bit-stream transmitting mode”	
the SIO status when the interrupt occurs	proper action
If SIO interrupt is generated, it means the SIO interface is waiting for data to be transmitted.	By writing data to the SIO_DATA register, the interrupt will happen after the SIO interface transmitting this data. Setting the interrupt clear register to clear interrupt



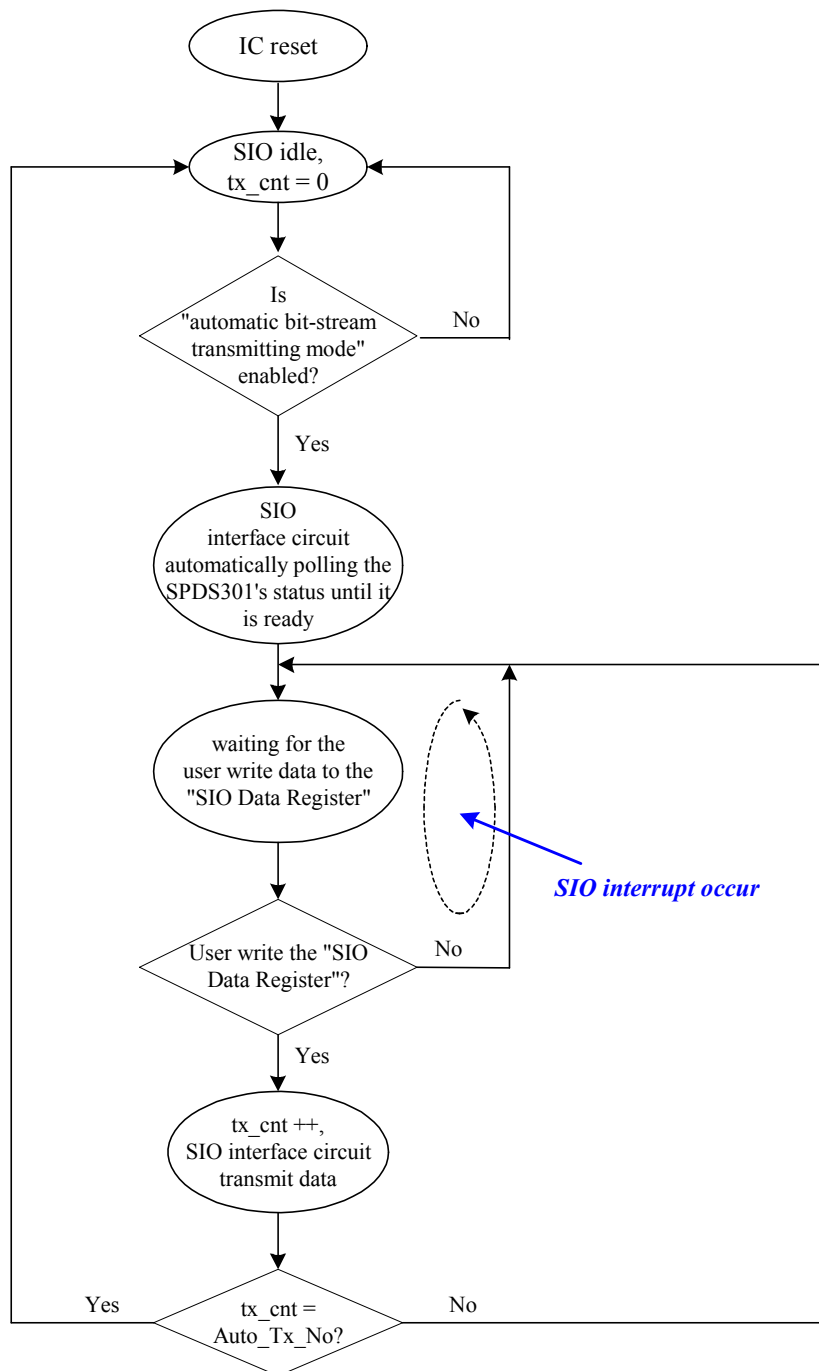
## 24.6 SIO Usage Procedure

(a) Not using “automatic bit-stream transmitting mode”





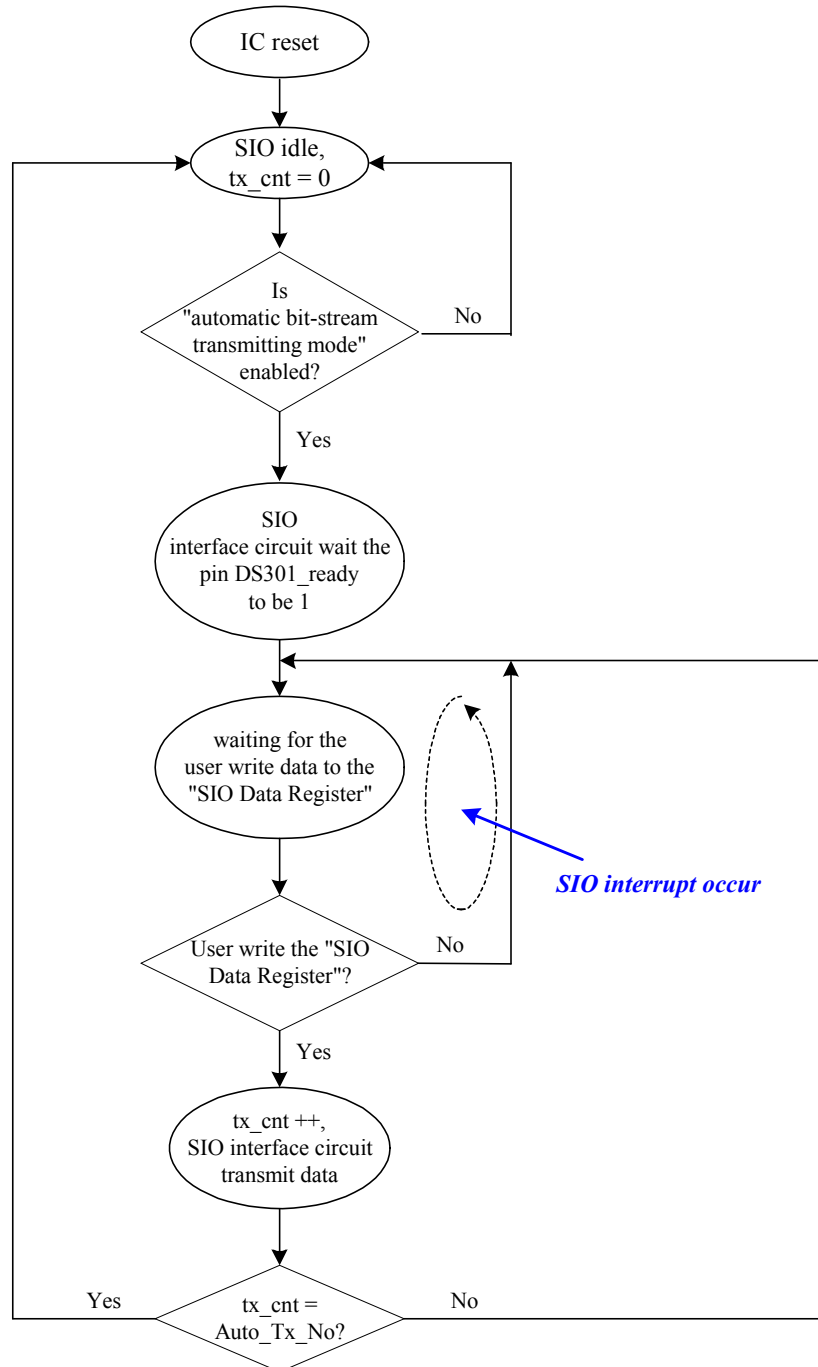
(b) Using "automatic bit-stream transmitting mode" & **not** using DS301\_ready pin







(c) Using both "automatic bit-stream transmitting mode" & DS301\_ready pin



---

## 25 SERIAL PERIPHERAL INTERFACE - SPI

---

### 25.1 General Description

The SPI performs parallel-to-serial conversion on data written to an internal 8-bit wide transmit buffer. It also performs serial-to-parallel conversion on the serial input data, and buffers it in a receiving buffer that is also 8-bits wide. The SPI asserts interrupts to request service to the transmitting buffer and to the receiving buffer, and to indicate an overrun condition in the receiving buffer.

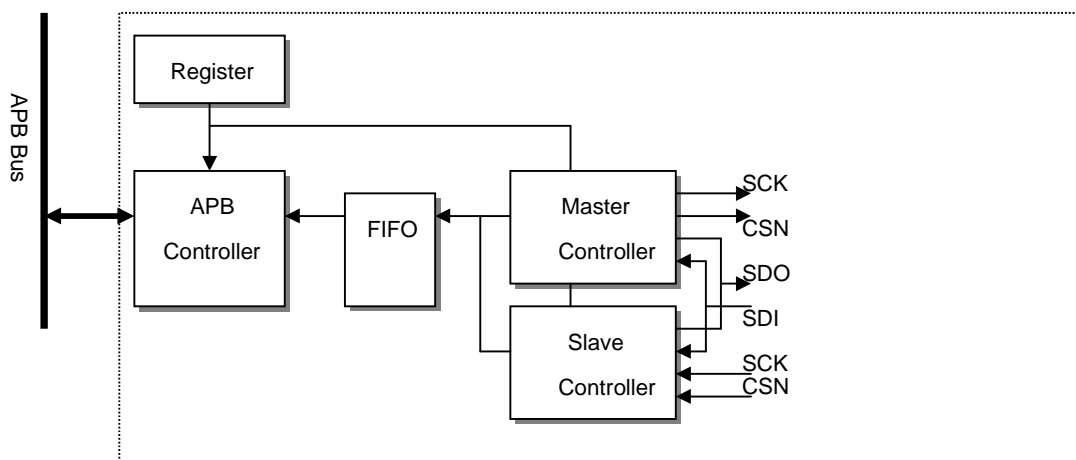
The SPI includes a programmable bit rate clock divider and pre-scalar to generate the serial output clock SCLK from the internal clock. The operating mode, frame format and size are programmed using control register P\_SPI\_Ctrl. The SSb output operates as an active LOW slave selection for SPI and Micro wire.

### 25.2 Features

A Serial Peripheral Interface (SPI) controller is built in SPG290 to facilitate communicating with other devices and components. There are four control signals on SPI, such as SPICSN, SPICLK, SPIRX, SPITX. There are some main features are as follow:

- 1). Support Master / Slave mode for single byte and consecutive bytes transferring.
- 2). Support overrun error indication
- 3). Support transmitting / receiving interrupt request
- 4). Programmable phase and polarity of master clock
- 5). Selectable data sampling time
- 6). Programmable master SCK clock frequency (System clock /2, /4, /8, /16, /32, /64, /128)
- 7). Built-in 8-depth 8bits FIFO in both transmit and receive direction, the interrupt level of those two FIFOs is programmable.

### 25.3 Architecture



### 25.4 Control Registers.

#### 25.4.1 P\_SPI\_Control(0x88110000)

NAME	D31	D30-D28	D27	D26	D25	D24-D8	D7	D6	D5-D3	D2-D0
P_SPI_Control	SPI_EN		SPI_LB	SOFT_REST	SPI_MODE		SPI_CK_PH	SPI_CK_PO		SPI_CLOCK

SPI\_CLOCK: Master Clock Select

- 000: PCLK/2
- 001: PCLK/4
- 010: PCLK/8
- 011: PCLK/16
- 100: PCLK/32
- 101: PCLK/64
- 110: PCLK/128

SPI\_CK\_PO: Refer to timing diagram

SPI\_CK\_PH: Refer to timing diagram

SPI\_MODE: SPI Master/Slave Mode Select

- 0: Master
- 1: Slave

SOFT\_REST: SPI Software Reset

- 0: No effect
- 1: Reset

SOFT\_LB: SPI Loop Back Mode Select(Master)

- 0: Normal
- 1: SPIRX= SPITX

SOFT\_EN: SPI Enable

0: Disable

1: Enable

#### 25.4.2 P\_SPI\_TX\_Status (0x88110004)

NAME	D31	D30	D29-D28	D27	D26-D7	D6-D4	D3	D2-D0
P_SPI_TX_Status	TX_FLAG	TX_EN		TX_EMPTY_FLAG		TX_FIFO_INT_LEVEL		FIFO_DATA_LEVEL

FIFO\_DATA\_LEVEL: SPI Tx FIFO Data Level

000: data number < 1, interrupt will happen

001: data number < 2, interrupt will happen

010: data number < 3, interrupt will happen

011: data number < 4, interrupt will happen

100: data number < 5, interrupt will happen

101: data number < 6, interrupt will happen

110: data number < 7, interrupt will happen

111: data number < 8, interrupt will happen

TX\_EMPTY\_FLAG: SPI Tx FIFO Empty Flag

0: No-Empty

1: Empty

TX\_EN: SPI Tx Interrupt Enable

0: Disable

1: Enable

TX\_FLAG: SPI Tx Interrupt Flag

Read 0: No happen

Read 1: Interrupt happened

Write 1: Clear Flag

#### 25.4.3 P\_SPI\_TX\_DATA (0x88110008)

NAME	D7-D0
P_SPI_TX_DATA	SPI_TX_DATA

#### 25.4.4 P\_SPI\_RX\_Status (0x8811000C)

NAME	D31	D30	D29-D28	D27	D26	D25-D5	D6-D4	D3	D2-D0
P_SPI_R_Status	R_FLAG	RX_EN		R_EMPTY_FLAG	RX_OR_ERR		R_FIFO_INT_LEVEL		FIFO_DATA_LEVEL

FIFO\_DATA\_LEVEL: SPI Rx FIFO Data Level

000: one data in FIFO, interrupt will happen

001: two data in FIFO, interrupt will happen

010: three data in FIFO, interrupt will happen



011: four data in FIFO, interrupt will happen  
 100: five data in FIFO, interrupt will happen  
 101: six data in FIFO, interrupt will happen  
 110: seven data in FIFO, interrupt will happen  
 111: eight data in FIFO, interrupt will happen

RX\_OR\_ERR: SPI Rx overrun error  
 Read 0: no happen  
 Read 1: happened  
 Write 1: clear flag

RX\_FULL\_FLAG: SPI Rx FIFO Full Flag  
 0: No-Full  
 1: Full

RX\_EN: SPI Rx Interrupt Enable  
 0: Disable  
 1: Enable

Rx\_FLAG: SPI Rx Interrupt Flag  
 Read 0: No happen  
 Read 1: Interrupt happened  
 Write 1: Clear Flag

#### 25.4.5 P\_SPI\_RX\_DATA (0x88110008)

NAME	D7-D0
P_SPI_RX_DATA	SPI_RX_DATA

#### 25.4.6 P\_SPI\_RX\_Status (0x8811000C)

NAME	D31	D30	D29-D5	D4	D3	D2	D1	D0
P_SPI_R_Status	SPI_OW_MODE	SPI_SMART		SPI_BUSY	RX_FULL	RX_EMPTY	TX_FULL	TX_EMPTY

TX\_EMPTY: SPI Transmit FIFO empty flag  
 0: No-Empty  
 1: Empty

TX\_FULL: SPI Transmit FIFO Full flag  
 0: full  
 1: not full

RX\_EMPTY: SPI receiver FIFO Empty flag  
 0: empty  
 1: no-empty

RX\_FULL: SPI receiver FIFO full flag  
 0: not full



	1:	full
SPI_BUSY:	SPI busy status	
	0:	Idle
	1:	Busy
SPI_SMART:	SPI smart mode	
	0:	Manual Clear IRQ
	1:	Auto Clear IRQ
SPI_OW_MODE:	SPI Overwrite mode	
	0:	skip data when OV
	1:	overwrite data when OV

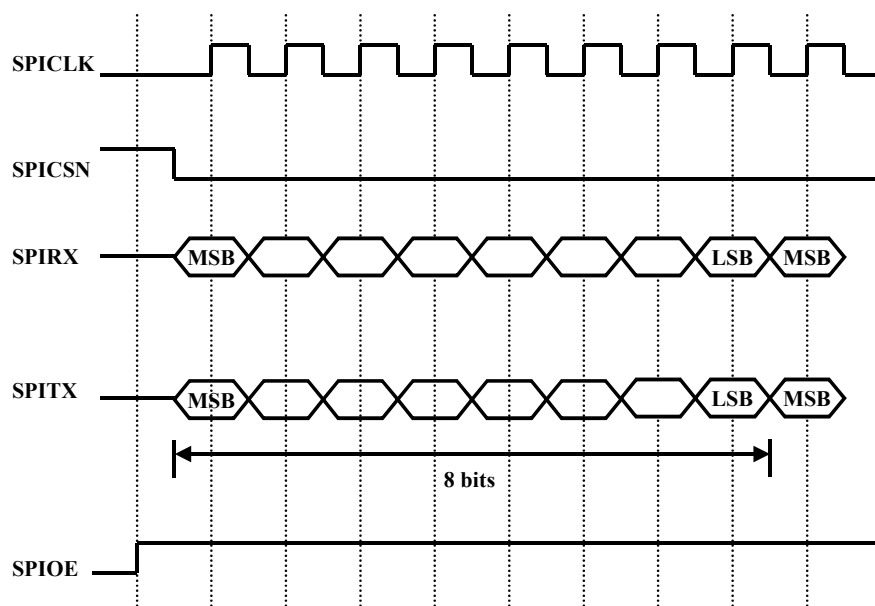


Fig1. Master Mode, SPO = 0, SPH=0

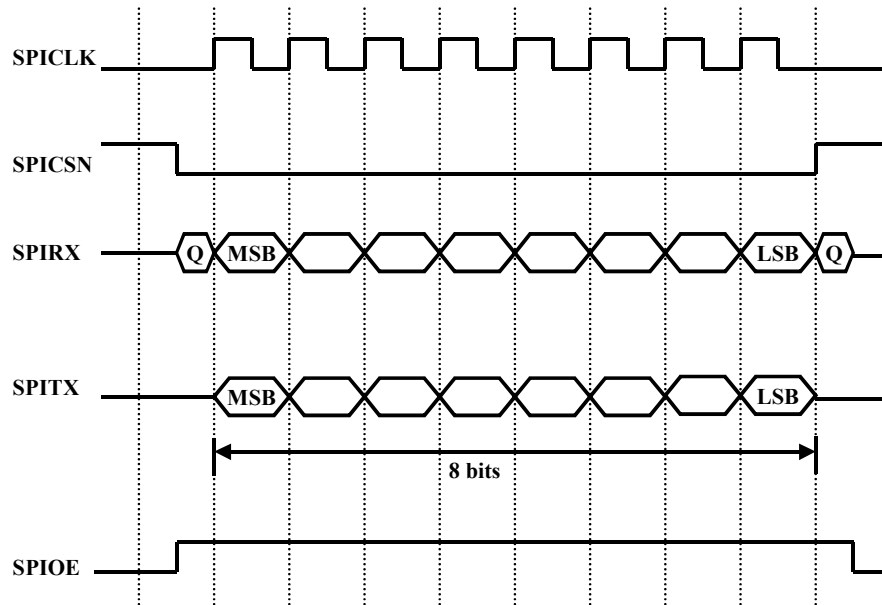


Fig2. Master Mode, SPO = 0, SPH=1

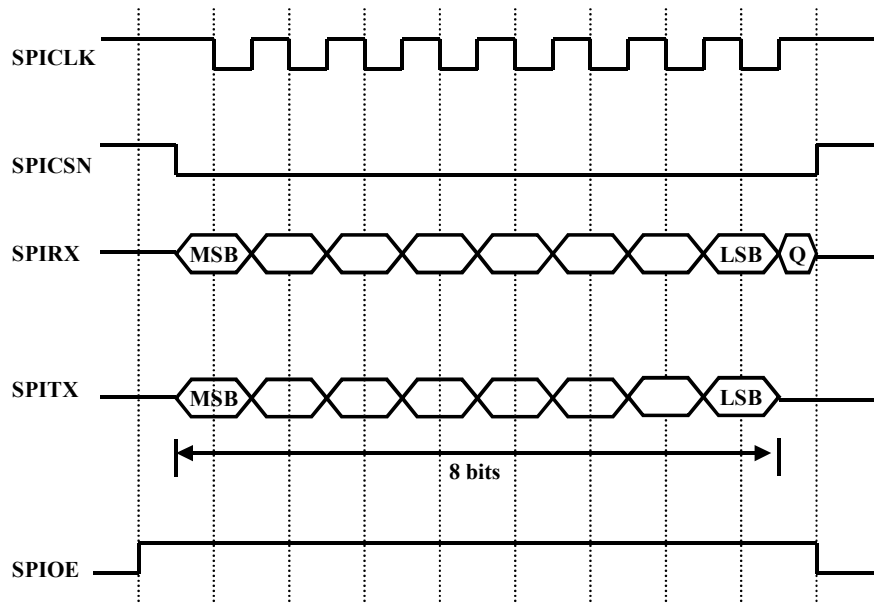


Fig3. Master Mode, SPO = 1, SPH=0

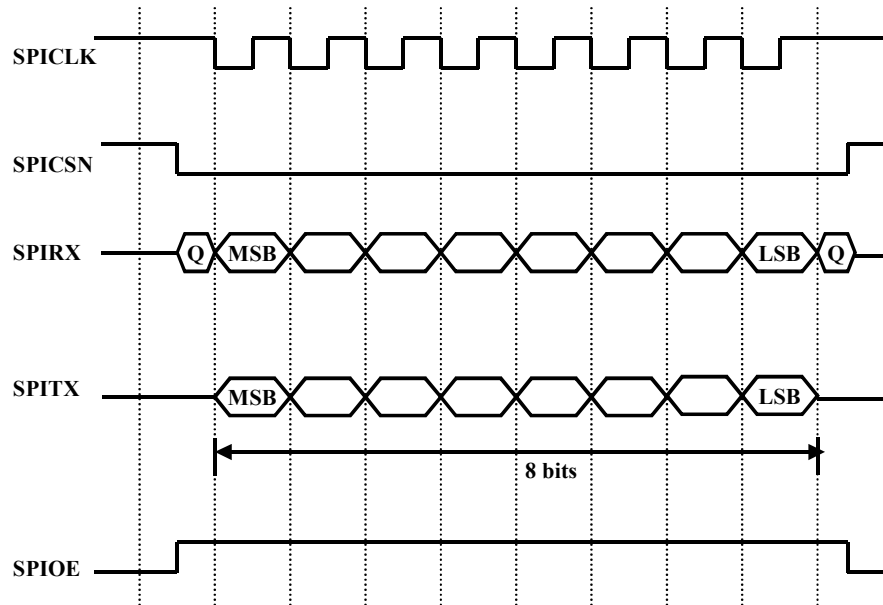


Fig4. Master Mode, SPO = 1, SPH=1



---

## 26 PERIPHERAL INTERRUPT CONTROLLER

---

### 26.1 INTRODUCTION

The interrupt controller provides masking capability for all of 40-interrupt sources and combines them into their final state for IRQ processor interrupt. The role of interrupt controller is to ask the IRQ interrupt request to the Score processor after making arbitration process when there are multiple interrupt requests from internal peripherals and external interrupt request pins.

Score processor only receives the IRQ interrupt events from the peripheral devices, but does not have a priority mechanism in it. So the software like interrupt handler must handle this kind of issues with hardware in1 interrupt controller. For example, assume that all of interrupt sources have been set as IRQ, and 10 of them requested interrupt to processor at same time, the software can determine the interrupt service priority by reading interrupt pending register, which denotes what kind of interrupt request happens.

This kind of interrupt process requires long interrupt latency to jump to the exact service routine. To resolve those inefficient processes, it provides a new interrupt processing mechanism called vectored interrupt mode, which is general feature of the CISC type Microprocessor. When the multiple interrupt sources request interrupt, hardware priority logic determines which interrupt should be serviced. At same time, this hardware logic applies the offset value from the vector table base address, which has a jump instruction to the corresponding service routine. So, by executing only several instructions to fetch a vector table entry with hardware assistance, the interrupt service can be executed. Compared with the software solution, it will reduce the interrupt latency dramatically.

### 26.2 FEATURE

- I Compliance to the AMBA specification(Rev 2.0) for easy integration into System-on-a-chip (SoC) implementation.
- I 40 interrupt sources.
- I Each interrupt is high active, and level sensitive IRQ.
- I Vectored interrupt support.
- I Programmable Priority of each interrupt source.

Figure 1. Functional Block Diagram of Interrupt Controller

### 26.3 Control Register

The following table shows the registers associated with the interrupt controller and physical address used to access them.

#### 26.3.1 P\_INTPND( R)(0x880A0000): INTPND Register.

Each of the 40 bits in the interrupt pending register, INTPND, corresponds to an interrupt source. When an interrupt request is generated from the external device, the corresponding bit in the register is set to "1". Although several interrupt sources generate requests simultaneously, the INTPND indicates what interrupt sources generate the interrupt requests.

NAME	D31-D0
P_INTPND	INTPND

INTPND: Indicates the interrupt request status

0: The interrupt has not been requested

1: The interrupt source has asserted the interrupt request

#### 26.3.2 P\_INTPND\_H( R)(0x880A0004):

NAME	D7-D0
P_INTPND_H	INTPND_H

INTPND\_H: Indicates the interrupt request status

0: The interrupt has not been requested

1: The interrupt source has asserted the interrupt request

#### 26.3.3 P\_I\_PMST(R/W)(0x880A0008)

This register determines the priority of among 4 IRQ service slave. Each IRQ service slave can accept 8 IRQ source.

NAME	D7-D0
P_I_PMST	X_PMST

X\_PMST: This 8 bits determines priorities among 4 slaves.

[1:0]: Priority of slave group 0

[3:2]: Priority of slave group 1

[5:4]: Priority of slave group 2

[7:6]: Priority of slave group 3

00 = 1<sup>st</sup>, 01 = 2<sup>nd</sup>, 02 = 3<sup>rd</sup>, 03 = 4<sup>th</sup>



### 26.3.4 P\_I\_GPUSWI(R/W)(0x880A000C)

NAME	D8	D7-D1	D0
P_I_GPUSWI	CLR_GPU_SWI		GPU_SWI

GPU\_SWI: GPU software interrupt bit

Write 1: Enter IRQ number 53

CLR\_GPU\_SWI: Clear interrupt flag

Write 1: Clear Software interrupt flag

### 26.3.5 IRQ Priority Register

This register determines the priority of each IRQ service slave. Each IRQ service slave can accept 8 IRQ source.

NAME	ADDR	D23-D21	D20-D18	D17-D15	D14-D12	D11-D9	D8-D6	D5-D3	D2-D0
P_I_PSLV0	0x880A0010	PSLV0_7	PSLV0_6	PSLV0_5	PSLV0_4	PSLV0_3	PSLV0_2	PSLV0_1	PSLV0_0
P_I_PSLV1	0x880A0014	PSLV1_7	PSLV1_6	PSLV1_5	PSLV1_4	PSLV1_3	PSLV1_2	PSLV1_1	PSLV1_0
P_I_PSLV2	0x880A0018	PSLV2_7	PSLV2_6	PSLV2_5	PSLV2_4	PSLV2_3	PSLV2_2	PSLV2_1	PSLV2_0
P_I_PSLV3	0x880A001C	PSLV3_7	PSLV3_6	PSLV3_5	PSLV3_4	PSLV3_3	PSLV3_2	PSLV3_1	PSLV3_0

PSLVX\_0: Priority of source 0

PSLVX\_1: Priority of source 1

PSLVX\_2: Priority of source 2

PSLVX\_3: Priority of source 3

PSLVX\_4: Priority of source 4

PSLVX\_5: Priority of source 5

PSLVX\_6: Priority of source 6

PSLVX\_7: Priority of source 7

NOTE: X = Slave X

000 = 1<sup>st</sup>, 001 = 2<sup>nd</sup>, 010 = 3<sup>rd</sup>, 011 = 4<sup>th</sup>

100 = 5<sup>th</sup>, 101 = 6<sup>th</sup>, 110 = 7<sup>th</sup>, 111 = 8<sup>th</sup>

## 26.4 VECADDR

The interrupt handler has the vector address to return to the processor respective to IRQ interrupt. The interrupt handler uses this vector value without calculating the offset address in the interrupt vector table to find the interrupt service routine. It allows the processor to skip the search process required in the non-vector mode to get the offset address whenever it receives the interrupt. This mechanism makes a system improve the overall performance dramatically. The value in this register will not change until another interrupt happens.

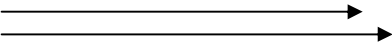


## 26.5 Interrupt Sources

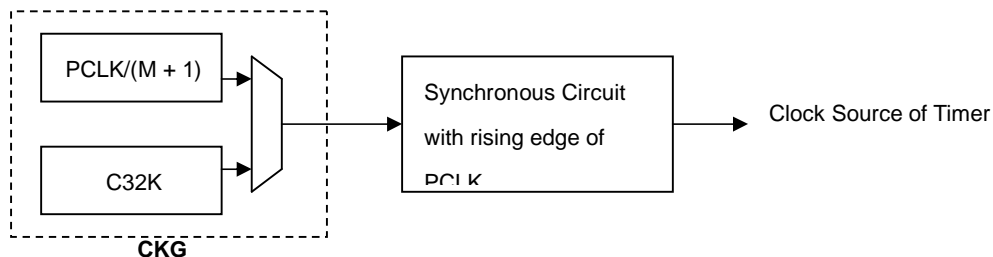
Slave Group	Source Number	Source	Vector address
0	0	SPU FIQ	63
	1	SPU Beatirq	62
	2	SPU Envirq	61
	3	CD servo	60
	4	ADC gain overflow / ADC recorder FIFO overflow	59
	5	General purpose ADC	58
	6	Timer base	57
	7	Timer	56
1	8	TV vblanking start	55
	9	LCD vblanking start	54
	10	PPU vblanking start	53
	11	TV 光槍	52
	12	Sensor frame end	51
	13	Sensor coordinate hit	50
	14	Sensor motion frame end	49
	15	Sensor capture done + sensor debug IRQ	48
2	16	TV coordinate hit	47
	17	PPU coordinate hit	46
	18	USB host+device	45
	19	SIO	44
	20	SPI	43
	21	Uart (IrDA)	42
	22	NAND	41
	23	SD	40
3	24	I2C master	39
	25	I2S slave	38
	26	APBDMA CH1	37
	27	APBDMA CH2	36
	28	LDM_DMA	35
	29	BLN_DMA	34
	30	APBDMA CH3	33
	31	APBDMA CH4	32
4	32	Alarm + HMS	31
	33	MP4	30
	34	C3 (ECC module)	29
	35	GPIO	28
	36	Bufctl (fordebug) + TV/PPU vblanking end (for debug)	27
	37	RESERVED1	26
	38	RESERVED2	25
	39	RESERVED3	24

## 27 TIMER

### 27.1 Features

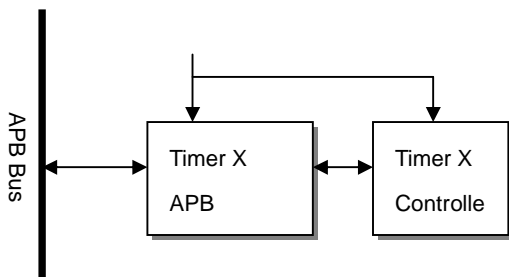
SPG290 contains six 16-bit timer/counters. All timers support Capture/Comparison/PWM (CCP) functions when cooperates with their own two 16-bit registers (preload register and ccp register). The clock source of these six timers can be programmed by internal clock (PCLK/2(max 13.5MHz) or C32K). Detail 

- 1). PCLK/2 base and C32K base clock source selection for each timer source
- 2). Each Programmable clock must be synchronous with PCLK as for each timer
- 3). Each timer support Capture function / Comparison function / PWM function



M = 1 ~ 255, and M = 0 is illegal

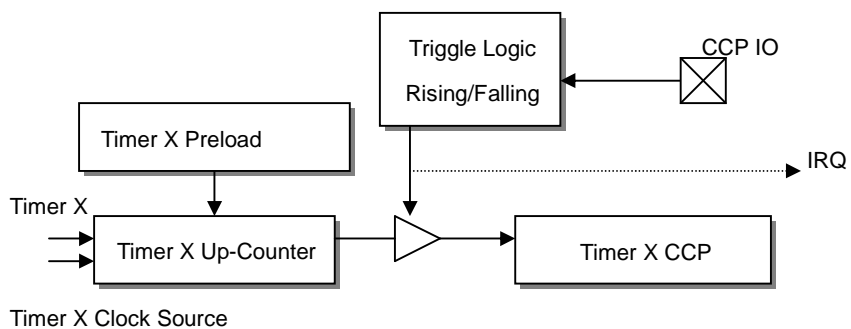
### 27.2 Architecture



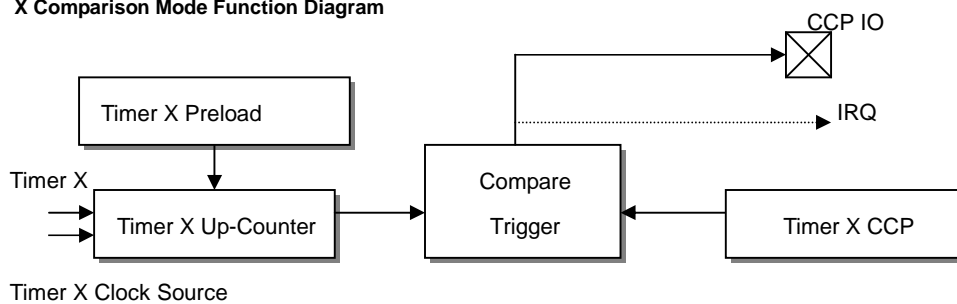
X = 0 ~ 5 (Timer 0 ~ Timer 5)



### X Capture Mode Function Diagram

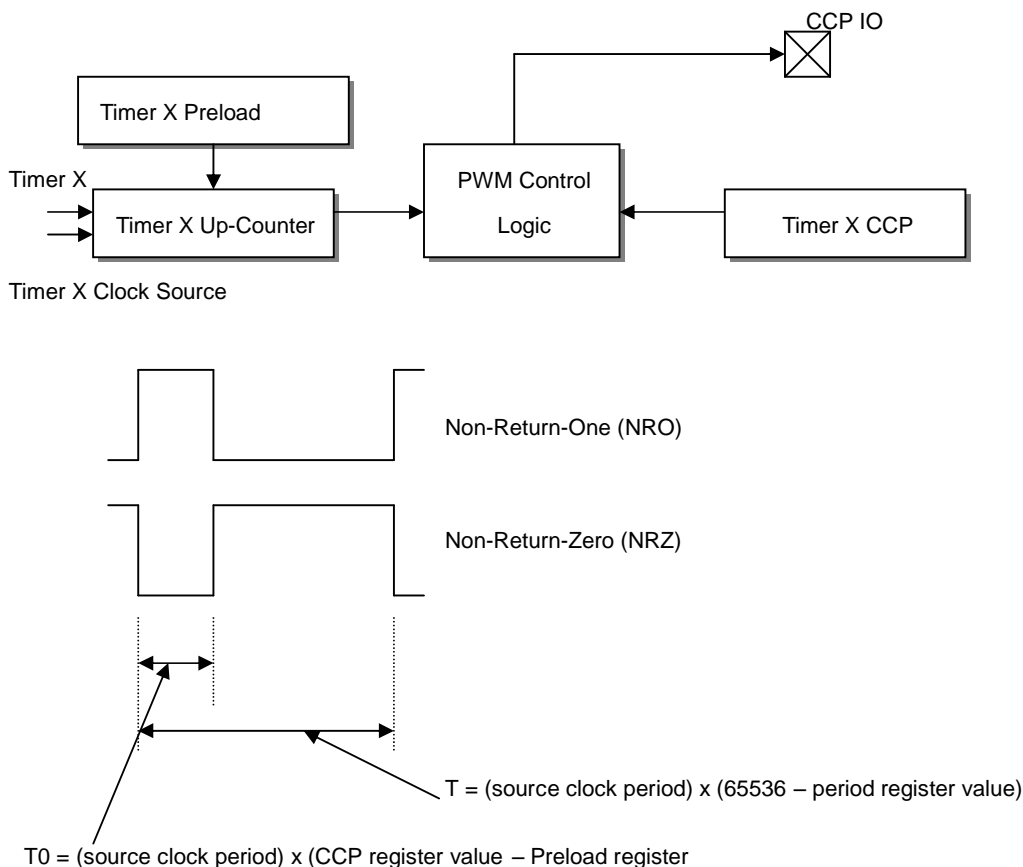


### X Comparison Mode Function Diagram





X PWM Mode Function Diagram



## 27.3 Control Registers

### 27.3.1 P\_TimerXCtrl(0x8816X000)

NAME	D31	D30-D28	D27	D26	D25-D0
P_TimerXCtrl	TIMER_X_EN		TIMER_X_IRQ_EN	TIMER_X_IRQ_FLAG	

TIMER\_X\_IRQ\_FLAG: Timer X Interrupt Flag

- 0: Disable
- 1: Enable

TIMER\_X\_IRQ\_EN: Timer X Interrupt Enable

- 0: Disable
- 1: Enable

TIMER\_X\_EN: Timer X Enable

- Read 0: Non Happen
- Read 1: irq occurred
- Write 1: Clear



### 27.3.2 P\_CPP\_CTRL(0x8816X004)

NAME	D31-D30	D29-D28	D27	D26	D25	D24-D0
P_CPP_CTRL	CCP_MODE		CAP_MODE	COM_MODE	PWM_MODE	

PWM\_MODE: PWM Mode Selection

0: NRO output

1: NRZ output

COM\_MODE: Comparison Mode Selection

0: high pulse

1: low pulse

CAP\_MODE: Capture Mode Selection

0: every falling

1: every rising

CCP\_MODE: CCP Mode Selection

00: Timer Mode

01: Capture Mode

10: Comparison Mode

11: PWM Mode

### 27.3.3 P\_Preload\_REGS(0x8816X008)

NAME	D31-D0
P_Preload_REGS	TIMER_X_PRELOAD_REGS

### 27.3.4 P\_CPP\_REGS(0x8816X00C)

NAME	D31-D0
P_CPP_REGS	TIMER_X_CCP_REGS

### 27.3.5 P\_CPP\_UPCOUNT(0x8816X010)

NAME	D31-D0
P_CPP_UPCOUNT	TIMER_X_UPCOUNT





---

---

## 28 RTC

---

---



---

---

## 29 WDOG

---

---



---

---

## 30 SLEEP/WAKEUP

---

---

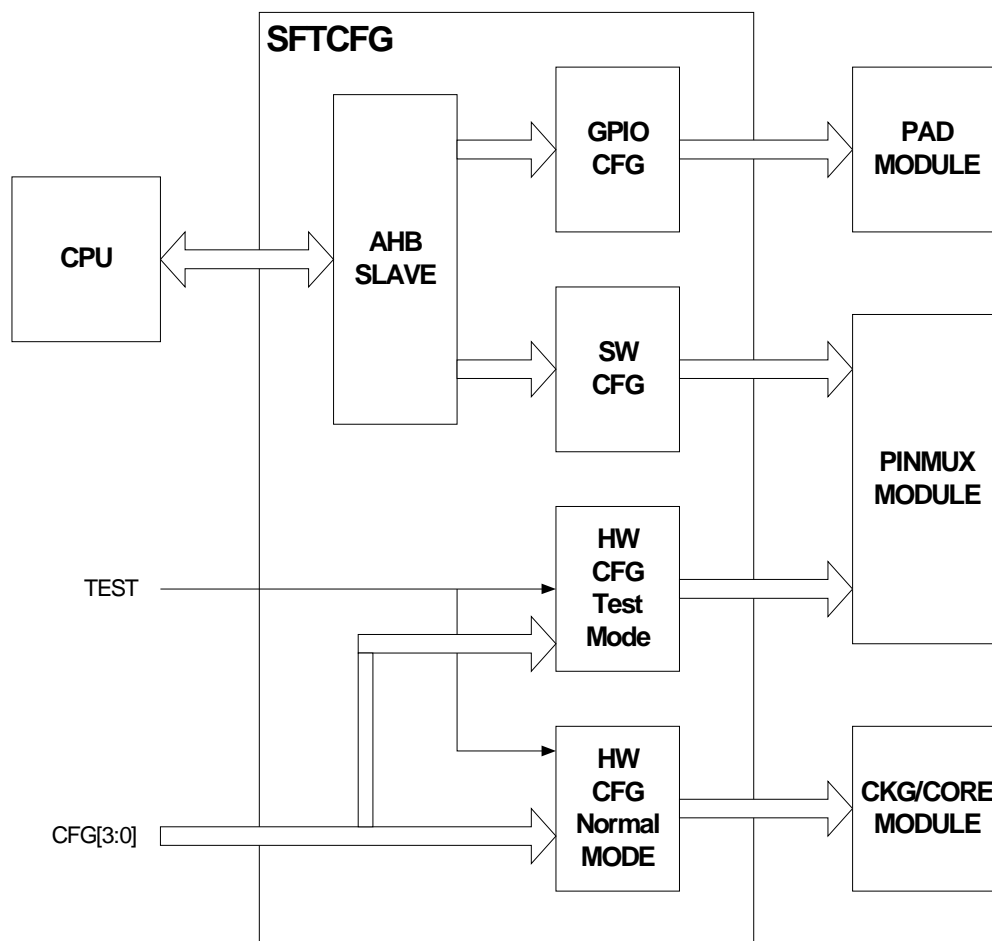
## 31 SOFTWARE CONFIG - SFTCFG

### 31.1 Function Description

There are the four group signals generated by the Software Configuration (SFTCFG) module. The first is the hardware configuration in the normal mode and generates the jtag enable, warm configuration reset and boot source signals. The second is the hardware configuration in the test mode and the generated signals are connected to the pin multiplex (PINMUX) module as the mode selection signals. The third is the software configuration by the cpu programming the registers and the generated signals are also connected to the pin multiplex (PINMUX) module as the mode selection signals. The fourth is the GPIO configuration by the cpu programming the registers and the generated signals including output port data, output enable, pull up and pull down signals are also connected to the input/output pads.

### 31.2 Architecture

The architecture of the SFTCFG module is shown below.





### 31.3 Control registers

#### 31.3.1 P\_GPIO\_DEVICE\_SET(0x88200000)

NAME	D24	D23-D17	D16	D15-D10	D9-D8	D7-D2	D1-D0
P_GPIO_DEVICE_SET	SW_UART		SW_USB		SW_CSI		SW_LCD

SW\_LCD: Software Configuration for LCD selection

- 00: TFT\_AUO
- 01: TFT\_TOPPOLY
- 10: STN
- 11: Reserved

SW\_CSI: Software Configuration for Sensor Interface Format selection

- 00: CCIR601
- 01: Sunplus Format
- 10: CCIR656
- 11: Reserved

SW\_USB: Software Configuration for USB Detect selection

- 0: As GPIO
- 1: USB Detect Selection

SW\_UART: Software Configuration for UART selection

- 0: As GPIO
- 1: UART Selection

#### 31.3.2 P\_GPIO\_FUNC\_SET(0x88200004)

NAME	D24	D23-D17	D16	D15-D10	D9-D8	D7-D1	D0
P_GPIO_FUNC_SET	SW_EROM		SW_CSI_PROBE		SW_PERI		SW_I2C

SW\_I2C: Software Configuration for I2C Selection

- 0: As GPIO
- 1: I2C Selection

SW\_PERI: Software Configuration for Peripheral Selection

- 00: JTAG
- 01: I2S
- 10: SIO
- 11: Reserved

SW\_CSI\_PROBE: Software Configuration for Sensor Hsync/Vsync/Field Probe Selection

- 0: Disable
- 1: Enable

SW\_EROM: Software Configuration for External ROM Selection

- 0: As GPIO



1: External ROM Selection

### 31.3.3 P\_GPIO\_DRAM\_SET(0x88200008)

NAME	D24	D23-D17	D16	D15-D9	D8	D7-D0
P_GPIO_DRAM_SET	SW_DRAM_BA1		SW_DRAM_A12		SW_DRAM_A11	

SW\_DRAM\_A11: Software Configuration for DRAM A11 Selection  
0: As GPIO  
1: DRAM A11 Selection

SW\_DRAM\_A12: Software Configuration for DRAM A12 Selection  
0: As GPIO  
1: DRAM A12 Selection

SW\_DRAM\_BA1: Software Configuration for DRAM BA1 Selection  
0: As GPIO  
1: DRAM BA1 Selection

### 31.3.4 P\_GPIO\_OUTPUT(0x8820000C)

NAME	D8	D0
P_GPIO_OUTPUT	SW_VDAC_OUT	SW_USBTRX_OUT

SW\_USBTRX\_OUT: Software Configuratiion for USB transceiver Output  
0: Disable  
1: Enable

SW\_VDAC\_OUT: Software Configuration for VIDEO DAC Output  
0: Disable  
1: Enable

### 31.3.5 P\_GPIO\_TFT\_PORT(0x88200010)

NAME	D19-D0
P_GPIO_TFT_PORT	TFT_GPIO_O

TFT\_GPIO\_O: TFT GPIO Port Data

### 31.3.6 P\_GPIO\_TFT\_OE(0x88200014)

NAME	D19-D0
P_GPIO_TFT_OE	TFT_GPIO_OE

TFT\_GPIO\_OE: TFT GPIO Port Output Enable  
0: Disable  
1: Enable

### 31.3.7 P\_GPIO\_TFT\_PU(0x88200018)

NAME	D19-D0
P_GPIO_TFT_PU	TFT_GPIO_PU

TFT\_GPIO\_PU: TFT GPIO Port Pull Up Enable  
0: Disable



1: Enable

### 31.3.8 P\_GPIO\_TFT\_PD(0x8820001C)

NAME	D19-D0
P_GPIO_TFT_PD	TFT_GPIO_PD

TFT\_GPIO\_PD: TFT GPIO Port Pull Down Enable

0: Disable

1: Enable

### 31.3.9 P\_GPIO\_CSI\_PORT(0x88200020)

NAME	D28-D16	D15-D13	D12-D0
P_GPIO_CSI_PORT	CSI_GPIO_OE		CSI_GPIO_O

CSI\_GPIO\_O: CSI GPIO Port Data

CSI\_GPIO\_OE: CSI GPIO Port Output Enable

0: Disable

1: Enable

### 31.3.10 P\_GPIO\_CSI\_PULL(0x88200024)

NAME	D28-D16	D15-D13	D12-D0
P_GPIO_CSI_PULL	CSI_GPIO_PD		CSI_GPIO_PU

CSI\_GPIO\_PU: CSI GPIO Port Pull Up Enable

0: Disable

1: Enable

CSI\_GPIO\_PD: CSI GPIO Port Pull Down Enable

0: Disable

1: Enable

### 31.3.11 P\_GPIO\_NFLASH\_PORT(0x88200028)

NAME	D31-D16	D15-D0
P_GPIO_NFLASH_PORT	NFLASH_GPIO_OE	NFLASH_GPIO_O

NFLASH\_GPIO\_O: NFLASH GPIO Port Data

NFLASH\_GPIO\_OE: NFLASH GPIO Port Output Enable

0: Disable

1: Enable

### 31.3.12 P\_GPIO\_NFLASH\_PULL(0x8820002C)

NAME	D31-D16	D15-D0
P_GPIO_NFLASH_PULL	NFLASH_GPIO_PD	NFLASH_GPIO_PU

NFLASH\_GPIO\_PU: NFLASH GPIO Port Pull Up Enable

0: Disable

1: Enable

NFLASH\_GPIO\_PD: NFLASH GPIO Port Pull Down Enable

0: Disable

1: Enable



### 31.3.13 P\_GPIO\_JTAG\_PORT(0x88200030)

NAME	D28-D24	D23-D21	D20-D16	D15-D13	D12-D8	D7-D5	D4-D0
P_GPIO_JTAG_PORT	JTAG_GPIO_PD		JTAG_GPIO_PU		JTAG_GPIO_OE		JTAG_GPIO_O

JTAG\_GPIO\_O: JTAG GPIO Port Data

JTAG\_GPIO\_OE: JTAG GPIO Port Output Enable

0: Disable

1: Enable

JTAG\_GPIO\_PU: JTAG GPIO Port Pull Up Enable

0: Disable

1: Enable

JTAG\_GPIO\_PD: JTAG GPIO Port Pull Down Enable

0: Disable

1: Enable

### 31.3.14 P\_GPIO\_GLOBAL\_PORT(0x88200034)

NAME	D25-D24	D23-D18	D17-D16	D15-D10	D9-D8	D7-D2	D1-D0
P_GPIO_GLOBAL_PORT	XGPIO_PD		XGPIO_PU		GLOBAL_OE		GLOBAL_O

GLOBAL\_O: GLOBAL GPIO Port Data

GLOBAL\_OE: GLOBAL GPIO Port Output Enable

0: Disable

1: Enable

XGPIO\_PU: GLOBAL GPIO Port Pull Up Enable

0: Disable

1: Enable

XGPIO\_PD: GLOBAL GPIO Port Pull Down Enable

0: Disable

1: Enable

### 31.3.15 P\_GPIO\_USB\_PORT(0x88200038)

NAME	D24	D23-D17	D16	D15-D9	D8	D7-D1	D0
P_GPIO_USB_PORT	XUSB_PD		XUSB_PU		USBDET_OE		USBDET_O

USBDET\_O: USBDET GPIO Port Data

USBDET\_OE: USBDET GPIO Port Output Enable

0: Disable

1: Enable

XUSB\_PU: USBDET GPIO Port Pull Up Enable

0: Disable

1: Enable

XUSB\_PD: USBDET GPIO Port Pull Down Enable

0: Disable

1: Enable

### 31.3.16 P\_GPIO\_UART\_PORT(0x8820003C)

NAME	D25-D24	D23-D18	D17-D16	D15-D10	D9-D8	D7-D2	D1-D0
P_GPIO_UART_PORT	XUART_PD		XUART_PU		UART_OE		UART_O





UART\_O: UART GPIO Port Data

UART\_OE: UART GPIO Port Output Enable  
0: Disable  
1: Enable

XUART\_PU: UART GPIO Port Pull Up Enable  
0: Disable  
1: Enable

XUART\_PD: UART GPIO Port Pull Down Enable  
0: Disable  
1: Enable

### 31.3.17 P\_GPIO\_I2C\_PORT(0x88200040)

NAME	D25-D24	D23-D18	D17-D16	D15-D10	D9-D8	D7-D2	D1-D0
P_GPIO_I2C_PORT	X I2C_PD		X I2C_PU		I2C_OE		I2C_O

I2C\_O: I2C GPIO Port Data

I2C\_OE: I2C GPIO Port Output Enable  
0: Disable  
1: Enable

XI2C\_PU: I2C GPIO Port Pull Up Enable  
0: Disable  
1: Enable

XI2C\_PD: I2C GPIO Port Pull Down Enable  
0: Disable  
1: Enable

### 31.3.18 P\_GPIO\_ADC\_PORT(0x88200044)

NAME	D31-D24	D23-D16	D15-D8	D7-D0
P_GPIO_ADC_PORT	XADC_PD	XAD_PU	ADC_OE	ADC_O

ADC\_O: ADC GPIO Port Data

ADC\_OE: ADC GPIO Port Output Enable  
0: Disable  
1: Enable

XADC\_PU: ADC GPIO Port Pull Up Enable  
0: Disable  
1: Enable

XADC\_PD: ADC GPIO Port Pull Down Enable  
0: Disable  
1: Enable

### 31.3.19 P\_GPIO\_DRAM\_PORT(0x88200048)

NAME	D27-D24	D23-D18	D19-D16	D15-D12	D11-D8	D7-D4	D3-D0
P_GPIO_DRAM_PORT	XDRAM_PD		XDRAM_PU		DRAM_OE		DRAM_O

DRAM\_O: DRAM GPIO Port Data

DRAM\_OE: DRAM GPIO Port Output Enable  
0: Disable



1: Enable  
XDRAM\_PU: DRAM GPIO Port Pull Up Enable  
0: Disable  
1: Enable  
XDRAM\_PD: DRAM GPIO Port Pull Down Enable  
0: Disable  
1: Enable

### 31.3.20 P\_GPIO\_ADC\_AEN(0x8820004C)

NAME	D7-D0
P_GPIO_ADC_AEN	ADC_GPIO_AEN

ADC\_GPIO\_AEN: ADC GPIO Port Analog Input Enable  
0: Disable  
1: Enable



---

---

## 32 JPEG

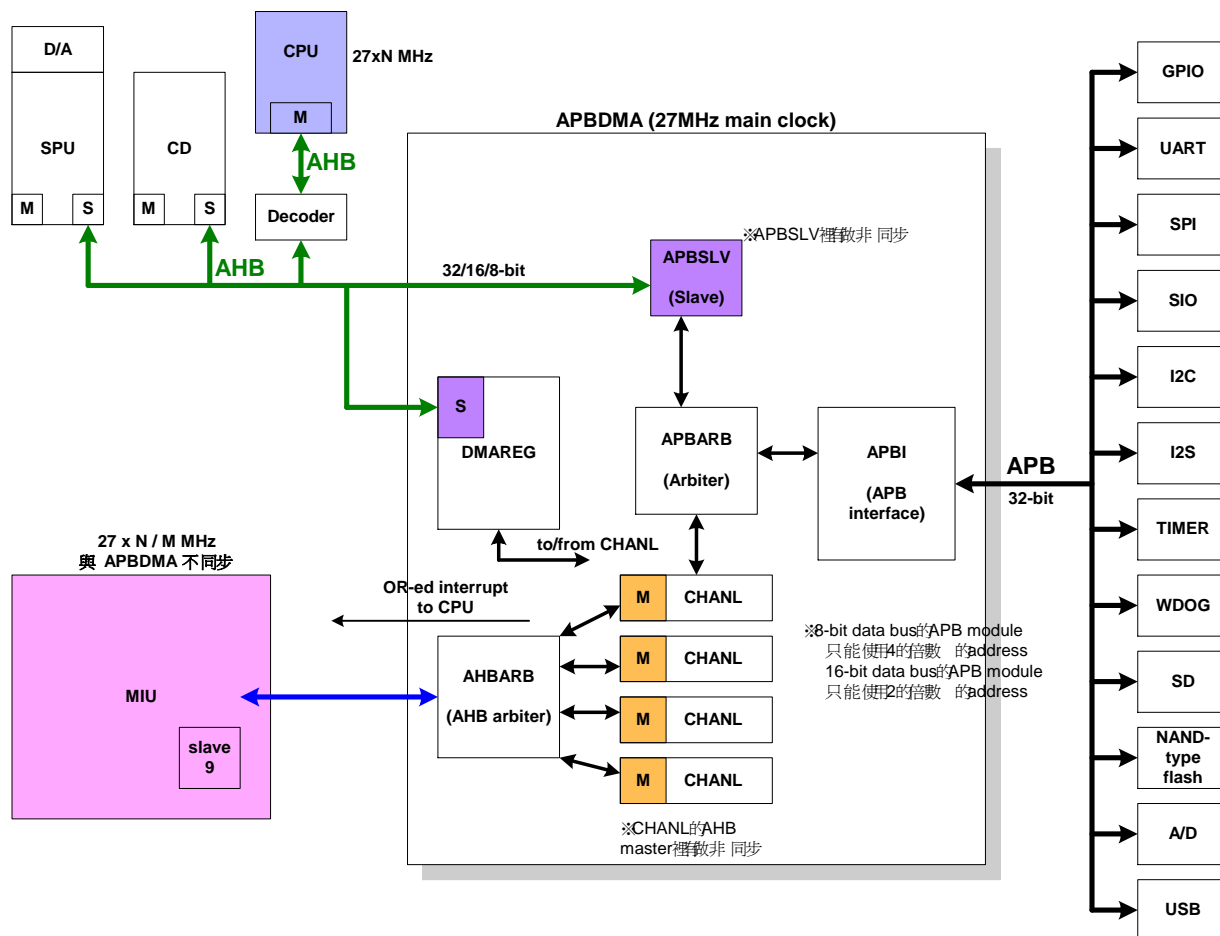
---

---

## 33 APBDMA

### 33.1 Architecture

### APB Bridge and DMA Controller:



## 33.2 APB Bridge

CPU can read and write each APB peripheral Module through APB Bridge

### 33.3 DMA Controller

DMA controller can offer four Channel to R/W MIU memory for APB peripheral modules at the same time, each one channel can be established into four kinds of transmission means:

- 1) 8 bits single transfer
- 2) 16 bits single transfer
- 3) 32 bits single transfer

4) 32 bits burst transfer.

DMA has two kinds of start modes:

- Case 1: When channel enable is set as 1, DMA controller begins successive R/W of the materials, until the content of the materials is R/W.
- Case 2: Acts as channel enable is set 1, and DMA controller carries out once R/W movements when APB peripheral modules send out REQ, over until all REQ finish sending DMA.

The space planning of Memory has two ways when DMA R/W MIU:

- Case 1: Single buffer establishes BUFFER A initial/stop position only, DMA R/W to this district intercropping only, will send out IRQ and stop DMA when R/W finishes.
- Case 2: Double buffer establishes BUFFER A/Buffer B initial/stop position at the same time, DMA is it R/W and finishes sending out IRQ, if should stop DMA to set channel enable as 0, treat DMA to will just send out IRQ and stop DMA after a block is finished and R/W entirely.

There are two kinds of DMA to that APB peripheral modules R/W making location ways of PORT:

- Case 1: Regular address
- Case 2: Continuous address

### 33.4 Control registers

#### 33.4.1 P\_CH\_BUSY\_STS(0x88080000)

NAME	D3	D2	D1	D0
P_CH_BUSY_STS	CH3_BUSY	CH2_BUSY	CH1_BUSY	CH0_BUSY

CH0\_BUSY: Channel 0 Busy Status

CH1\_BUSY: Channel 1 Busy Status

CH2\_BUSY: Channel 2 Busy Status

CH3\_BUSY: Channel 3 Busy Status

#### 33.4.2 P\_CH\_IRQ\_STS(0x88080004)

NAME	D3	D2	D1	D0
P_CH_IRQ_STS	CH3_IRQ	CH2_IRQ	CH1_IRQ	CH0_IRQ

CH0\_IRQ: Channel 0 IRQ Status

Read 0: DMA IRQ not occurred

Read 1: DMA IRQ occurred

Write 0: No operation

Write 1: DMA IRQ Clear

CH1\_IRQ: Channel 1 IRQ Status



Read 0: DMA IRQ not occurred  
Read 1: DMA IRQ occurred  
Write 0: No operation  
Write 1: DMA IRQ Clear

CH2\_IRQ: Channel 2 IRQ Status  
Read 0: DMA IRQ not occurred  
Read 1: DMA IRQ occurred  
Write 0: No operation  
Write 1: DMA IRQ Clear

CH3\_IRQ: Channel 3 IRQ Status  
Read 0: DMA IRQ not occurred  
Read 1: DMA IRQ occurred  
Write 0: No operation  
Write 1: DMA IRQ Clear

#### 33.4.3 Buffer A Start Address

NAME	ADDRESS	D31-D0
P_AHB_START_ADDR1_A	08080008H	BUFFER A Start Address For Channel1
P_AHB_START_ADDR2_A	0808000CH	BUFFER A Start Address For Channel2
P_AHB_START_ADDR3_A	08080010H	BUFFER A Start Address For Channel3
P_AHB_START_ADDR4_A	08080014H	BUFFER A Start Address For Channel4

#### 33.4.4 Buffer A End Address

NAME	ADDRESS	D31-D0
P_AHB_END_ADDR1_A	08080018H	BUFFER A End Address For Channel1
P_AHB_END_ADDR2_A	0808001CH	BUFFER A End Address For Channel2
P_AHB_END_ADDR3_A	08080020H	BUFFER A End Address For Channel3
P_AHB_END_ADDR4_A	08080024H	BUFFER A End Address For Channel4

#### 33.4.5 APB Module Access Port Start Address

NAME	ADDRESS	D31-D0
P_APB_START_ADDR1	08080028H	APB Module Access Port Start Address For Channel1
P_APB_START_ADDR2	0808002CH	APB Module Access Port Start Address For Channel2
P_APB_START_ADDR3	08080030H	APB Module Access Port Start Address For Channel3
P_APB_START_ADDR4	08080034H	APB Module Access Port Start Address For Channel4

#### 33.4.6 Buffer B Start Address

NAME	ADDRESS	D31-D0
P_AHB_START_ADDR1_B	0808004CH	BUFFER B Start Address For Channel1



<b>P_AHB_START_ADDR2_B</b>	<b>08080050H</b>	BUFFER B Start Address For Channel2
<b>P_AHB_START_ADDR3_B</b>	<b>08080054H</b>	BUFFER B Start Address For Channel3
<b>P_AHB_START_ADDR4_B</b>	<b>08080058H</b>	BUFFER B Start Address For Channel4

#### 33.4.7 Buffer B End Address

NAME	ADDRESS	D31-D0
<b>P_AHB_END_ADDR1_B</b>	<b>0808005CH</b>	BUFFER B End Address For Channel1
<b>P_AHB_END_ADDR2_B</b>	<b>08080060H</b>	BUFFER B End Address For Channel2
<b>P_AHB_END_ADDR3_B</b>	<b>08080064H</b>	BUFFER B End Address For Channel3
<b>P_AHB_END_ADDR4_B</b>	<b>08080068H</b>	BUFFER B End Address For Channel4

#### 33.4.8 P\_CH1\_SETTING (0x8808006C)

NAME	D7	D6	D5	D4	D3	D2	D1	D0
<b>P_CH1_SETTING</b>	CH1_EN	CH1_IRQ	CH1_TRANS		CH1_MEM	CH1_MODE	CH1_DMA	CH1_DIR

CH1\_DIR: Channel1 direction for R/W

0: MIU TO APB

1: APB TO MIU

CH1\_DMA: Channel1 DMA Mode Selection

0: Auto mode

1: Polling mode

CH1\_MODE: Channel1 DMA's APB peripheral Modules location mode

0: Continuous mode

1: Regular mode

CH1\_MEM: Channel1 MIU Memory mode

0: Single Buffer

1: Double Buffer

CH1\_TRANS: Channel1 transfer mode

00: 8 bits single transfer

01: 16 bits single transfer

10: 32 bits single transfer

11: 32 bits burst transfer

CH1\_IRQ: Channel1 IRQ Mask

0: Disable DMA IRQ

1: Enable DMA IRQ

CH1\_EN: Channel1 Enable

0: Disable DMA

1: Enable DMA



### 33.4.9 P\_CH2\_SETTING (0x88080070)

NAME	D7	D6	D5	D4	D3	D2	D1	D0
P_CH2_SETTING	CH2_EN	CH2_IRQ	CH2_TRANS	CH2_MEM	CH2_MODE	CH2_DMA	CH2_DIR	

CH2\_DIR: Channel2 direction for R/W  
0: MIU TO APB  
1: APB TO MIU

CH2\_DMA: Channel2 DMA Mode Selection  
0: Auto mode  
1: Polling mode

CH2\_MODE: Channel2 DMA's APB peripheral Modules location mode  
0: Continuous mode  
1: Regular mode

CH2\_MEM: Channel2 MIU Memory mode  
0: Single Buffer  
1: Double Buffer

CH2\_TRANS: Channel2 transfer mode  
00: 8 bits single transfer  
01: 16 bits single transfer  
10: 32 bits single transfer  
11: 32 bits burst transfer

CH2\_IRQ: Channel2 IRQ Mask  
0: Disable DMA IRQ  
1: Enable DMA IRQ

CH2\_EN: Channel2 Enable  
0: Disable DMA  
1: Enable DMA

### 33.4.10 P\_CH3\_SETTING (0x88080074)

NAME	D7	D6	D5	D4	D3	D2	D1	D0
P_CH3_SETTING	CH3_EN	CH3_IRQ	CH3_TRANS	CH3_MEM	CH3_MODE	CH3_DMA	CH3_DIR	

CH3\_DIR: Channel3 direction for R/W  
0: MIU TO APB  
1: APB TO MIU

CH3\_DMA: Channel3 DMA Mode Selection  
0: Auto mode  
1: Polling mode

CH3\_MODE: Channel3 DMA's APB peripheral Modules location mode  
0: Continuous mode





1: Regular mode

CH3\_MEM: Channel3 MIU Memory mode

0: Single Buffer

1: Double Buffer

CH3\_TRANS: Channel3 transfer mode

00: 8 bits single transfer

01: 16 bits single transfer

10: 32 bits single transfer

11: 32 bits burst transfer

CH3\_IRQ: Channel3 IRQ Mask

0: Disable DMA IRQ

1: Enable DMA IRQ

CH3\_EN: Channel3 Enable

0: Disable DMA

1: Enable DMA

#### 33.4.11 P\_CH4\_SETTING (0x88080078)

NAME	D7	D6	D5	D4	D3	D2	D1	D0
P_CH4_SETTING	CH4_EN	CH4_IRQ	CH4_TRANS	CH4_MEM	CH4_MODE	CH4_DMA	CH4_DIR	

CH4\_DIR: Channel4 direction for R/W

0: MIU TO APB

1: APB TO MIU

CH4\_DMA: Channel4 DMA Mode Selection

0: Auto mode

1: Polling mode

CH4\_MODE: Channel4 DMA's APB peripheral Modules location mode

0: Continuous mode

1: Regular mode

CH4\_MEM: Channel4 MIU Memory mode

0: Single Buffer

1: Double Buffer

CH4\_TRANS: Channel4 transfer mode

00: 8 bits single transfer

01: 16 bits single transfer

10: 32 bits single transfer

11: 32 bits burst transfer

CH4\_IRQ: Channel4 IRQ Mask



0: Disable DMA IRQ  
1: Enable DMA IRQ  
CH4\_EN: Channel4 Enable  
0: Disable DMA  
1: Enable DMA

#### 33.4.12 P\_CH\_SW\_RST (0x8808007C)

NAME	D3	D2	D1	D0
P_CH_SW_RST	CH3_REST	CH2_REST	CH1_REST	CH0_REST

CH0\_REST: Channel 0 Software Reset  
0: Disable  
1: Enable  
CH1\_REST: Channel 1 Software Reset  
0: Disable  
1: Enable  
CH2\_REST: Channel 2 Software Reset  
0: Disable  
1: Enable  
CH3\_REST: Channel 3 Software Reset  
0: Disable  
1: Enable

### 33.5 DMA END

When DMA finishes must write 1 to relative IRQ STATUS bit, that will clear DMA interrupt, and must write 0 to relative CHANNEL ENABLE, DMA that could start next time so deposit and withdraw.

## 34 NAND-TYPE FLASH

### 34.1 INTRODUCTION

There is a NAND-type flash controller embedded in SPG290. The NAND-type flash is suitable for mass storage application. The controller support standard 8-bits NAND flash equipped with ECC calculation circuit. The smart media card is also supported.

### 34.2 FEATURE

- Fully programmable CLE and ALE sequences which can support varies kind of NAND flash.
- Support polling/interrupt/DMA access method.
- Support page size from 528 bytes to 2112 bytes.
- Hardware ECC calculation circuit.
- Programmable read/write cycles.
- Programmable ALE cycles.

### 34.3 BLOCK DIAGRAM

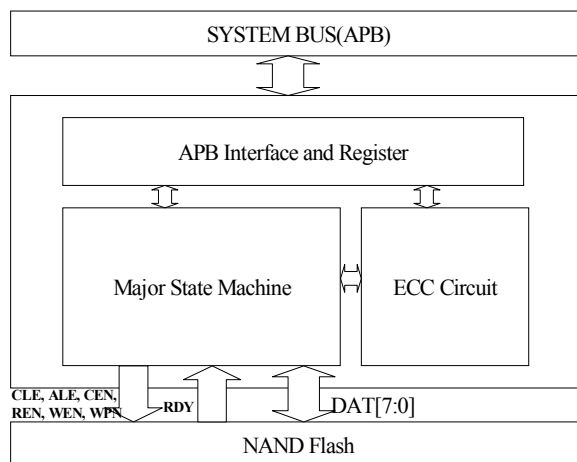


Figure 1. Functional Block Diagram of Flash Controller

### 34.4 INTERFACE SIGNAL

#### 34.4.1 APB Bus Signals

This part is the same as define in AMBA specification.



#### 34.4.2 Flash Bus Signals

CEN	O	Chip select of NAND flash, low active
CLE	O	Command latch enable, high active
ALE	O	Address latch enable, high active
WEN	O	Write enable, low active
REN	O	Read enable, low active
WPN	O	Write protect enable, low active
RDY	I	NAND flash ready input, high active
DATA[7:0]	I/O	Data bus

#### 34.4.3 Other Signals

INT	O	Interrupt pin, high active
DMARQ	O	DMA request signal, high active

### 34.5 Control Registers

#### 34.5.1 P\_FL\_CR(R/W)(0x8800F000): Control Register

Flash controller control register. This register is used to control the behavior of controller. For example, the timing setting of page size setting. Programming must program this register based on the specification of NAND flash. Host uses this register to control the behavior of the controller.

NAME	D7-D6	D5-D4	D3	D2	D1	D0
P_FL_CR	ALECYC	RDTYPE	WRCMD	FLWPN	FLCEN	FLEN
NAME	D31-D28	D27-D16	D15-D14	D13-D12	D11-D10	D9-D8
P_FL_CR		TBYTES	RDHIGH	RDLOW	WRHIGH	WRLOW

FLEN: Flash interface enable register.

0: Disable

1: Enable

FLCEN: CEN control register.

0: CEN output 0

1: CEN output 1

FLWPN: WPN control register.

0: WPN output 0

1: WPN output 1

WRCMD: Read/Write command control

0: Read Command

1: Write Command

RDTYPE:	Indicate when to start the read process
00:	Read after RDY rise
01:	Read after CLE fall
10:	Read after ALE fall
11:	Reserved
ALECYC:	Indicate how many writes will be issued when P_FL_ALE is written
00:	One write cycle
01:	Two write cycles
10:	Three write cycles
11:	Four write cycles
WRLOW:	Indicate the clock cycles of WEN low pulse
00:	One Clock cycle
01:	Two Clock cycle
10:	Three Clock cycles
11:	Four Clock cycles
WRHIGH:	Indicate the clock cycles of WEN high pulse.
00:	One clock cycle.
01:	Two clock cycles.
10:	Three clock cycles.
11:	Four clock cycles.
RDLOW:	Indicate the clock cycles of REN low pulse.
00:	One clock cycle.
01:	Two clock cycles.
10:	Three clock cycles.
11:	Four clock cycles.
RDHIGH:	Indicate the clock cycles of REN high pulse.
00:	One clock cycle.
01:	Two clock cycles.
10:	Three clock cycles.
11:	Four clock cycles.
TBYTES:	Total bytes -1 of this transfer

### 34.5.2 P\_FL\_CLE(R/W)(0x8800F004): Command Latch enable register

This register is used to control the CLE of NAND flash bus. When programmer write data to this register when controller is not busy, a CLE pulse will be issued with command data written in this register.

NAME	D7-D0
P_FL_CLE	FL_CLE

FL\_CLE: When Programmer write data to this register when controller is not busy, a CLE pulse will be issued with command data written in this register

### 34.5.3 P\_FL\_ALE(R/W)(0x8800F008): Address latch enable register

This register is used to control the ALE of NAND flash bus. When programmer write data to this register when controller is not busy, a ALE pulse will be issued with address data written in this register. The cycles of ALE command is defined in FL\_CR register.

NAME	D31-D0
P_FL_ALE	FL_ALE

FL\_ALE: When programmer write data to this register when controller is not busy, a ALE pulse will be issued with command data written in this register.

### 34.5.4 P\_FL\_WD(R/W)(0x8800F00C): Write data register

This register is used to write data to NAND flash bus. Programmer must write data to this register when WREMPY is 1. Otherwise, the WRLOSS bit will be set and the written data will loss.

NAME	D31-D0
P_FL_WD	FL_WD

FL\_WD: This register is used to write data to NAND flash bus. Programmer must write data to this register when WREMPY is 1.

### 34.5.5 P\_FL\_RD(R/W)(0x8800F010): Read data register

This register is used to read data from NAND flash bus. Programmer must read data from this register when RDFULL is 1. Otherwise, the RDMISS bit will be set and the read data is invalid.

NAME	D31-D0
P_FL_RD	FL_RD

FL\_RD: This register is used to read data from NAND flash bus. Programmer must read data from this register when RDFULL is 1.



### 34.5.6 P\_FL\_INTEN(R/W)(0x8800F014): Interrupt enable register

Host uses this register to control the interrupt output.

NAME	D5	D4	D3	D2	D1	D0
P_FL_INTEN	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0

INTEN0: Write Empty Interrupt enable select

0: Disable

1: Enable

INTEN1: Read Full Interrupt enable select

0: Disable

1: Enable

INTEN2: Ready Rise Interrupt enable select

0: Disable

1: Enable

INTEN3: Write Loss Interrupt enable select

0: Disable

1: Enable

INTEN4: Read Miss Interrupt enable select

0: Disable

1: Enable

INTEN5: Command Loss Interrupt enable select

0: Disable

1: Enable

### 34.5.7 P\_FL\_INTSTS(R/C)(0x8800F018): Interrupt status register

The register is used to read-out the status of flash controller.

NAME	D7	D6	D5	D4	D3	D2	D1	D0
P_FL_INTSTS	RDY	BUSY	CMDLOSS	RDMISS	WRLOSS	RDYRIS	RDFULL	WREMP

WREMP: Write Empty Interrupt status

Read 0: Write buffer is full, further write to P\_FL\_WD will cause Write Loss become 1, and the write data will loss.

Read 1: Write buffer is empty, further write is allowed.

**NOTE:** This bit will be clear automatically when the write buffer is written.

RD\_FULL: Read Full Interrupt status

Read 0: Read buffer is empty, further read from P\_FL\_RD will cause Read Miss become 1, and the read data is invalid.

Read 1: Read buffer is full, a read from FL\_RD is allowed.

**NOTE:** This bit will be clear automatically when the read buffer is read.



RDYRISE:	Ready Rise interrupt status
Read 0:	RDY rise is not happen.
Read 1:	RDY rise is happen.
Write 0:	No effect.
Write 1:	Clear this bit.
WRLOSS:	Write Loss Interrupt status
Read 0:	Write loss event not happen.
Read 1:	Write loss event happen.
Write 0:	No effect.
Write 1:	Clear this bit.
RDMISS:	Read Miss Interrupt status
Read 0:	Read miss event not happen.
Read 1:	Read miss event happen.
Write 0:	No effect.
Write 1:	Clear this bit.
CMDLOSS:	Command Loss Interrupt status
Read 0:	Command loss event not happen.
Read 1:	Command loss miss event happen.
Write 0:	No effect.
Write 1:	Clear this bit.
BUSY:	Controllers busy flag
0:	Controller is not busy, ALE/CLE command is allowed.
1:	Controller is busy, ALE/CLE command is not allowed. If a ALE/CLE command is received in this state, CMDLOSS will be set to 1 and the command will loss.
RDY:	NAND flash busy flag, this bits will shows the status of RDY on the NAND flash bus.
0:	NAND flash is busy.
1:	NAND flash is not busy.

#### 34.5.8 P\_FL\_TRUELP(R/W)(0x8800F01C): True line parity register

Programmer can use this register to write the true LP result for ECC circuit to calculate the error bits in a page.

NAME	D31-D16	D15-D0
P_FL_TRUELP	TRUELP1	TRUELP0

TRUELP0: The true line parity of bytes 0~255 in a page. This register is written by programmer and used in error line/bit calculation



TRUELP1: The true line parity of bytes 256~511 in a page. This register is written by programmer and used in error line/bit calculation.

#### 34.5.9 P\_FL\_TRUECP(R/W)(0x8800F020): True column parity register

Programmer can use this register to write the true CP result for ECC circuit to calculate the error bits in a page.

NAME	D11-D6	D5-D0
P_FL_TRUECP	TRUECP1	TRUECP0

TRUECP0: The true column parity of bytes 0~255 in a page. This register is written by programmer and used in error line/bit calculation.

TRUECP1: The true column parity of bytes 256~511 in a page. This register is written by programmer and used in error line/bit calculation.

#### 34.5.10 P\_FL\_CALLP(R)(0x8800F024): Calculate line a parity register

Programmer can read the line parity of ECC calculation result from this register.

NAME	D31-D16	D15-D0
P_FL_CALLP	CALLP1	CALLP0

CALLP0: The line parity of bytes 0~255 in a page calculated by ECC circuit. This register is written by ECC circuit and used in error line/bit calculation.

CALLP1: The line parity of bytes 256~511 in a page calculated by ECC circuit. This register is written by ECC circuit and used in error line/bit calculation.

#### 34.5.11 P\_FL\_CALCP(R)(0x8800F028): Calculate column parity register

Programmer can read the column parity of ECC calculation result from this register.

NAME	D11-D6	D5-D0
P_FL_CALCP	CALCP1	CALCP0

CALCP0: The column parity of bytes 0~255 in a page calculated by ECC circuit. This register is written by ECC circuit and used in error line/bit calculation.

CALCP1: The column parity of bytes 256~511 in a page calculated by ECC circuit. This register is written by ECC circuit and used in error line/bit calculation.



### 34.5.12 P\_FL\_ECCSTS(R)(0x8800F02C): ECC status register

The error line and error bit will be shown on this register.

NAME	D31-D30	D29-D28	D26-D24	D23-D16	D15-D14	D13-D11	D10-D8	D7-D0
P_FL_ECCSTS	ERR1		ERRBIT1	ERRBYT E1	ERR0		ERRBIT0	ERRBYT E0

ERRBYTE0: The error byte of bytes 0~255 detected by ECC circuit

ERRBIT0: The error bit of ERRBYTE0 detected by ECC circuit.

ERR0: Error detection of byte 0~255 in a page.

00: No error is detected

01: One error is detected, correctable.

10: Reserved

11: Two or more errors is detected, non-correctable.

ERRBYTE1: The error byte of bytes 256~511 detected by ECC circuit

ERRBIT1: The error bit of ERRBYTE1 detected by ECC circuit.

ERR1: Error detection of byte 256~511 in a page.

00: No error is detected

01: One error is detected, correctable.

10: Reserved

11: Two or more errors is detected, non-correctable.

### 34.5.13 P\_FL\_CALECC(W)(0x8800F030): ECC control register

This register is used to control the ECC circuit.

NAME	D2	D1	D0
P_FL_CALECC	CLRECC1	CLRECC0	CALECC

CALECC: Write 0: No effect.

Write 1: The ECC circuit will use FL\_TRUECP/FL\_TRUECP and  
FL\_CALLP/FL\_CALLP to find the error byte and error bit.

This bit will be clear after the calculation is done.

CLRECC0: Write 0: No effect.

Write 1: The ECC part 0 will be reset.

This bit will be clear after the calculation is done. This bit is used when the page  
size is larger than 528 bytes.

CLRECC1: Write 0: No effect.

Write 1: The ECC part 1 will be reset.

This bit will be clear after the calculation is done. This bit is used when the page  
size is larger than 528 bytes.

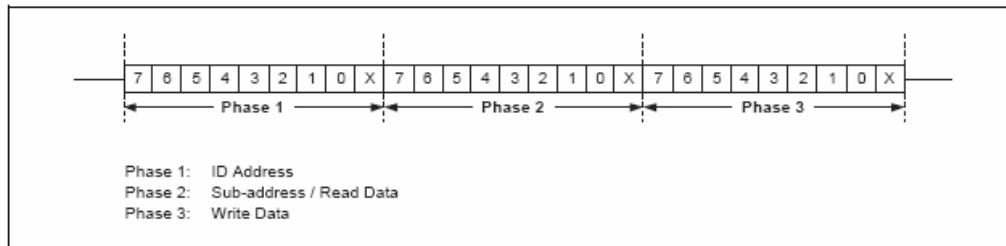
## 35 INTER-INTEGRATED CIRCUIT - I2C

### 35.1 Initialization

Before starting I2C, initialization must be done. There are 2 major jobs:

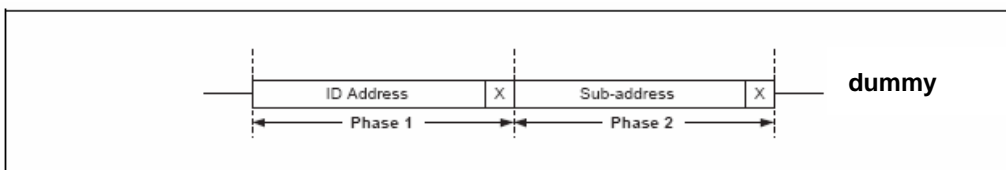
- (1) Decide the I2C clock rate by configuring I2CCVR
- (2) Decide the I2C slave ID Address by configuring I2CID

The SPG290 support I2C master only and 3 types of transaction : 8 bits type, 16 bits type and 8 bits \* n type. For 8 bits type: by writing I2CCR[0] to start the transaction and by reading I2CCR[3] to watch the result (if INTEN is 1, I2C interrupt happens after a full transaction). Here is an example for I2C 8 bits writing:

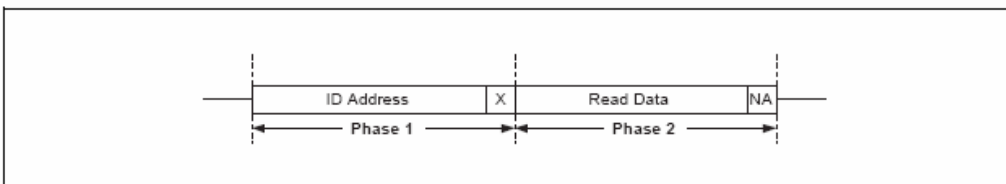


0<sup>th</sup> bit in ID Address must be 0

Besides, here is an example for I2C 8 bits reading:

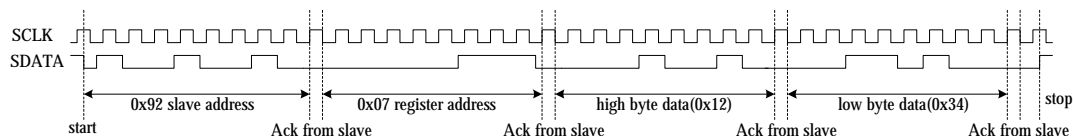


0<sup>th</sup> bit in ID Address must be 0

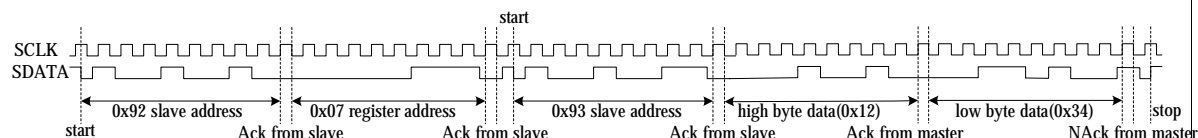


0<sup>th</sup> bit in ID Address must be 1

For 16 bits type : by writing I2CCR[1] to start the transaction and by reading I2CCR[4] to watch the result (if INTEN is 1, I2C interrupt happens after a full transaction). Here is an example for I2C 16 bits writing:



Here is an example for I2C 16 bits reading:



For 8 bits \* n type transaction : each transaction is looked as each phase transaction. The definition of “phase transaction” is presented as above. By writing I2CCR[2] to start each phase transaction and by reading I2CCR[5] to watch the result (if INTEN is 1, I2C interrupt happened after each phase transaction). In addition, by writing I2CCR[8] to stop the full transaction.

## 35.2 Control Registers

### 35.2.1 P\_I2C\_CR(R/W)(0x88130020): I2C configuration register.

NAME	D8	D7	D6	D5	D4	D3	D2	D1	D0
P_I2C_CR	STOPN	GAK	TRANS_MODE	ACKN	ACK16	ACK8	STARTN	START16	START8

START8:	Start 8 bits transfer
START16:	Start 16 bits transfer
STARTN:	Start general transaction
ACK8:	ACK from 8 bits transfer
ACK16:	ACK from 16 bits transfer
ACKN:	ACK from general transaction
TRANS_MODE:	Read/ Write Mode selection
	0: Write Mode
	1: Read Mode
GAK:	ACK/NAK mode select
	0: Generate NAK at last transaction
	1: Generate ACK at last transaction
STOPN:	Stop general transaction



**35.2.2 P\_I2C\_INTR(R/W)(0x88130024): I2C interrupt register.**

NAME	D1	D0
P_I2C_INTR	INT_EN	INT_FLAG

INT\_FLAG: Interrupt flag

INT\_EN: Interrupt enable bit

**35.2.3 P\_I2C\_CVR(R/W)(0x88130028): I2C clock setting register.**

NAME	D9-D0
P_I2C_CVR	CLK_SET

**35.2.4 P\_I2C\_ID(R/W)(0x8813002C): I2C ID register.**

NAME	D7-D1
P_I2C_ID	ID_REG

**35.2.5 P\_I2C\_ADDR(R/W)(0x88130030): I2C port address register.**

NAME	D7-D0
P_I2C_ADDR	ADDR

**35.2.6 P\_I2C\_WDATA(R/W)(0x88130034): I2C write data register.**

NAME	D15-D0
P_I2C_WDATA	I2C_WDATA

**35.2.7 P\_I2C\_RDATA(R/W)(0x88130038): I2C read data register.**

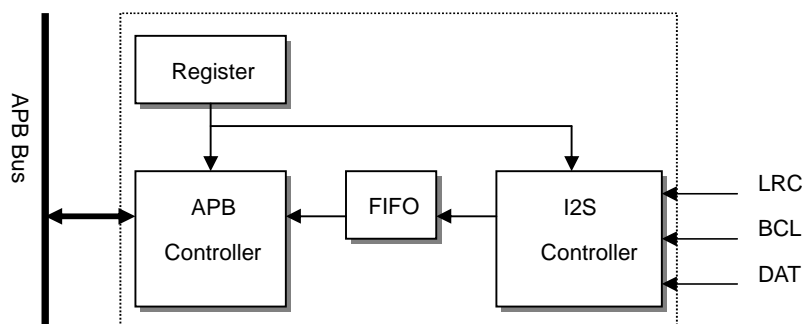
NAME	D15-D0
P_I2C_RDATA	I2C_RDATA

## 36 INTERACTIVE INFORMATION SERVICES - I2S

### 36.1 Features

This I2S controller only support for the receive function in the slave mode. The maximum receive data size is 32-bit for left/right channel.

### 36.2 Architecture



### 36.3 Control Registers

#### 36.3.1 P\_I2S\_CR(R/W)(0x88140000): I2S control

NAME	D31	D30-D28	D27	D26	D25	D24	D23	D22	D21	D20	D19-D0
P_I2S_CR	RX_EN		LJM	LCE	RCE	USB	LRF	DMA	FIFO	AUTO	

AUTO: Auto Interrupt flag clear for IRQ mode

0: Disable

1: Enable

FIFO: FIFO receive overwrite enable

0: Disable

1: Enable

DMA: DMA Enable

0: IRQ mode

1: DMA mode

LRF: Left / Right first receive select

0: Left

1: Right

USB: USB Mode

0: Disable

1: Enable

RCE: Right channel enable



0: Disable  
1: Enable  
LCE: Left channel enable  
0: Disable  
1: Enable  
LJM: I2S / Left justified mode select  
0: I2S mode  
1: Left mode  
RX\_EN: I2S Receiver Enable  
0: Disable  
1: Enable

### 36.3.2 P\_I2S\_IRQ\_STS(R/W)(0x88140004): I2S IRQ Status

NAME	D31	D30	D29-D0
P_I2S_IRQ_STS	INT_FLAG	INT_EN	

INT\_EN: I2S interrupt enable  
0: Disable  
1: Enable  
INT\_FLAG: I2S interrupt flag  
Read 1: IRQ occurred  
Write 1: Clear IRQ

### 36.3.3 P\_I2S\_FIFO(R/W)(0x88140008): I2S FIFO Config

NAME	D31	D30	D29	D28-D6	D5-D4	D3-D2	D1-D0
P_I2S_FIFO	OV_ERR	FULL_FLAG	EMPTY_FLAG		RX_INT_LV		RX_LV

RX\_LV: FIFO receiver level  
RX\_INT\_LV: FIFO receive interrupt level  
00: one receive will irq  
01: two receive will irq  
10: three receive will irq  
11: four receive will irq  
EMPTY\_FLAG: FIFO empty flag  
0: not empty  
1: empty  
FULL\_FLAG: FIFO full flag  
0: not full  
1: full  
OV\_ERR: FIFO receive overflow error



Read 1: Error occurred

Write 1: Clear

#### 36.3.4 P\_I2S\_RDATA(R/W)(0x8814000C): I2S receive data

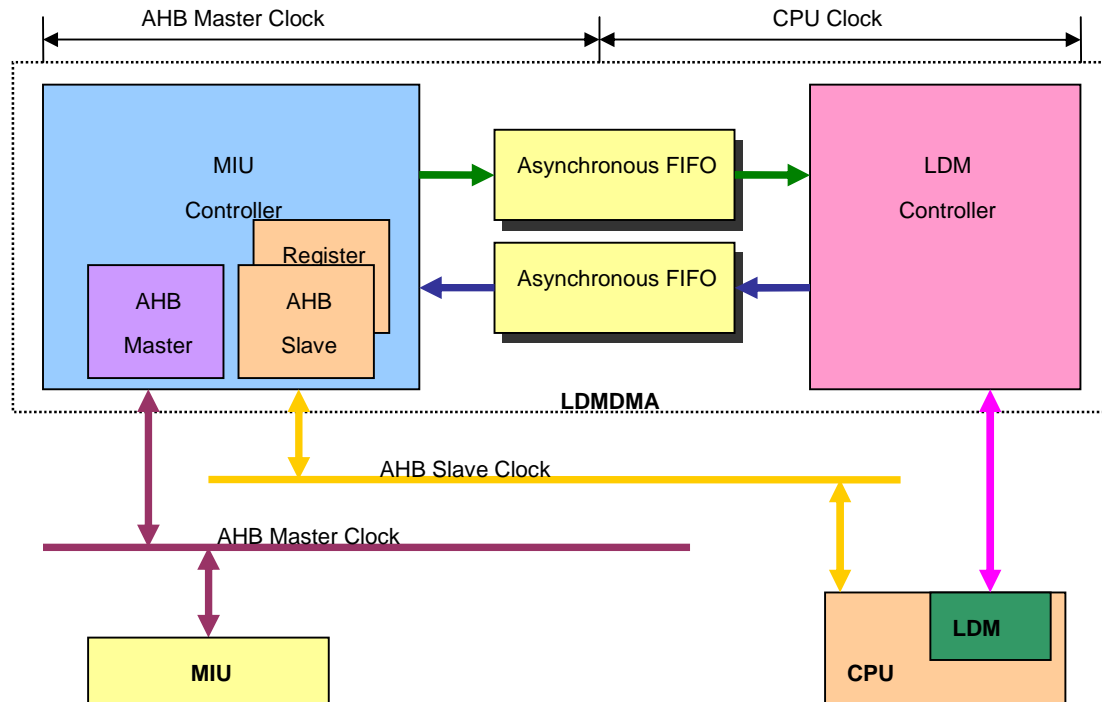
NAME	D31-D0
P_I2S_RDATA	I2S_RDATA





## 37 LOCAL DATA MEMORY - LDM

### 37.1 Architecture



The book is acted as AHB Master in order to deposit and withdraw MIU by MIU controller, LDM controller comes to act as master in order to deposit and withdraw Local Data Memory(LDM) within CPU, and established register within LDMDMA by CPU by AHB slave.

### 37.2 LDMDMA Controller

This DMA controller is half a duplexing way, only permit a kind of route(MIU to LDM/ LDM to MIU) to exist at the same time, When LDMDMA finishes, that will send out interrupt to notify CPU. There are four kinds of transmission ways that we support:

- 1) byte mode
- 2) half-word mode
- 3) word mode
- 4) 4-beat burst word mode

Because the whole system is that 32 bits bus in a main fact, that takes word address as the core, so will get better performance by 4-beat burst word mode or word mode. If want to use the transmission relying mainly on byte address or half-word address systematically at dawn, then we support byte mode and half-word mode too. So we have these four ways to receive better choice for users.



### 37.3 Control registers

Control registers in LDMDMA is written AHB slave into/read by CPU.

#### 37.3.1 P\_DMA\_CTRL(0x880C0000)

NAME	D31	D30	D29	D28	D27-D26	D25-D0
P_DMA_CTRL	MODULE_EN	IRQ_EN	DIRECT		TRANS_MODE	

TRANS\_MODE: Module enable

0: Disable

1: Enable

DIRECT: Access direction

0: MIU 2 LDM

1: LDM 2 MIU

IRQ\_EN: IRQ Enable

0: Disable

1: Enable

MODULE\_EN: Module enable

0: Disable

1: Enable

#### 37.3.2 P\_DMA\_STATUS(0x880C0004)

NAME	D31	D30-D0
P_DMA_STATUS	INT_REQ	

INT\_REQ: Interrupt request(IRQ) (MIN 2 LDM/ LDM 2 MIU finish)

Read 0: not occurred

Read 1: occurred

Write 0: no operation

Write 1: Clear Status

#### 37.3.3 P\_MIU\_START\_ADDR(0x880C0008)

NAME	D27-D0
P_MIU_START_ADDR	MIU_START_ADDR(Byte address mode)

#### 37.3.4 P\_MIU\_END\_ADDR(0x880C000C)

NAME	D27-D0
P_MIU_END_ADDR	MIU_END_ADDR(Byte address mode)

#### 37.3.5 P\_LDM\_START\_ADDR(0x880C0010)

NAME	D17-D0
P_LDM_START_ADDR	LDM_START_ADDR(Byte address mode)



**37.3.6 P\_LDM\_END\_ADDR(0x880C0014)**

NAME	D17-D0
P_LDM_END_ADDR	LDM_END_ADDR(Byte address mode)

## 38 ADC

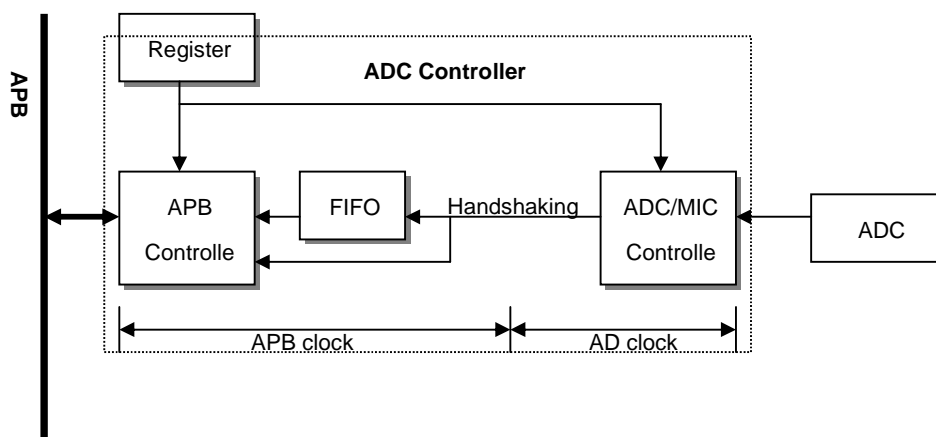
### 38.1 Features

The analog-to-digital converter provides voice record or general-purpose ad functions. There are total nine channels, and one channel dedicated for microphone mode and other channels dedicated for 12-bit general-purpose ad. The microphone can't support auto gain control(AGC), so we need additional software to support this function. And microphone data only use DMA to transfer instead of the interrupt mode. The general-purpose mode support auto data by sampled at programmable sampling rate or manual data by sampled at sampling command. And all general-purpose ADC only support interrupt mode to read data.

ADC spec. is as follows:

- 1). General-purpose application: 2.0mA
- 2). Audio application without MICBIAS: 4.0mA / with MICBIAS: 7.5mA
- 3). Power down mode: 1uA
- 4). Nine channels 12-bit resolution for audio and general-purpose ADC application
- 5). On-chip microphone boost amplifier and programmable gain amplifier(33, 31.5, ..., -12,  $-\infty$  db)
- 6). On-chip anti-aliasing filter and bias-voltage output(MICBIAS)
- 7). Reference top voltage and bias current/voltage generators

### 38.2 Architecture





### 38.3 Control Registers

#### 38.3.1 P\_ADC\_SETUP(0x881A0000): ADC Setup

NAME	D31	D30	D29	D28	D27	D26	D25	D24-D4	D3-D0
P_ADC_SETUP	ADC_SEL	ADC_EN	MIC_EN		MIC_BOOST	MIC_BIAS	TOP_VOL		CH_SEL

CH\_SEL:

Channel Select

0000: GPIO 0

0001: GPIO 1

0010: GPIO 2

0011: GPIO 3

0100: GPIO 4

0101: GPIO 5

0110: GPIO 6

0111: GPIO 7

1xxx: MIC

TOP\_VOL:

Top reference voltage internal or external

0: Internal

1: External

MIC\_BIAS:

MIC bias enable(only for MIC mode)

0: Disable

1: Enable

MIC\_BOOST:

MIC boost enable (only for MIC mode)

0: Disable

1: Enable

MIC\_EN:

MIC enable

0: Disable

1: Enable

ADC\_EN:

ADC enable

0: Disable

1: Enable

ADC\_SEL:

ADC Selection

0: Disable

1: Enable



### 38.3.2 P\_MIC\_GAIN(0x881A0004): MIC GAIN

NAME	D4-D0
P_MIC_GAIN	MIC_GAIN

MIC_GAIN:	MIC Gain value
00000:	33.0 db
00001:	31.5 db
00010:	30.0 db
.....	
10101:	1.5 db
10110:	0.0 db
10111:	-1.5 db
....	
11110:	-12.0 db
11111:	-∞

### 38.3.3 P\_ADC\_CLOCK\_SET(0x881A0008): ADC Clock Setup

NAME	D15-D0
P_ADC_CLOCK_SET	ADC_CLOCK_CYCLE

ADC\_CLOCK\_CYCLE: ([15:0] + 1) cycles ACC

### 38.3.4 P\_ADC\_HOLD\_SETUP(0x881A000C): Sample Hold Setup

NAME	D31-D28	D27-D16	D15-D0
P_ADC_HOLD_SETUP	HOLD_WIDTH		HOLD_CYCLE

HOLD\_CYCLE: ([15:0] + 1) cycles Sample Hold Cycle must  $\geq 15$

HOLD\_WIDTH: [3:0] width Sample Hold Width must  $\geq 2$

**NOTE:** The method about setup ADC clock cycle and sample-hold width / cycle  
If we need 44.1K sample rate for ADC and ADC controller clock = 33.8688 MHz  
Sample cycle =  $33.8688\text{M} / 44.1\text{K} = 768$ , and we need Sample Hold cycle = 16.  
Finally, ADC clock cycle =  $768 / 16 = 48$   
So ACC = 47, SampleHoldCycle = 15, and SampleHoldWidth = 2