HY16F3910 Series
Peripheral Driver
C Library

© 2021-2022 HYCON Technology Corp.

目錄

1. Overview 14
1.1. C Library Introduction 14
1.2. Relative Document 14
2. SYS Driver 15
2.1. Introudction 15
2.2. Functions 16
2.2.1. SYS_SleepFlagRead 16
2.2.2. SYS_SleepFlagClear 16
2.2.3. SYS_WdogFlagRead 16
2.2.4. SYS_WdogFlagClear 17
2.2.5. SYS_ResetFlagRead 17
2.2.6. SYS_ResetFlagClear 18
2.2.7. SYS_BOR_FlagRead 18
2.2.8. SYS_BOR_FlagClear 19
2.2.9. SYS_EnableGIE 19
2.2.10. SYS_DisableGIE 20
2.2.11. SYS_LowPower 20
2.2.12. SYS_INTPriority 21
3. CLOCK Driver 23
3.1. Introduction 23
3.2. Type Definition 24
3.3. Functions 25
3.3.1. DrvCLOCK_EnableHighOSC 25
3.3.2. DrvCLOCK_CloseEHOSC 25
3.3.3. DrvCLOCK_CloseIHOSC 26
3.3.4. DrvCLOCK_SelectIHOSC 26
3.3.5. DrvCLOCK_EnableLowOSC 27
3.3.6. DrvCLOCK_CloseELOSC 28
3.3.7. DrvCLOCK_SelectMCUClock 28
3.3.8. DrvCLOCK_TrimHAO 29
3.3.9. DrvCLOCK_CalibrateHAO 29
3.3.10. DrvCLOCK_SelectOHS_HS 30
3.3.11. DrvCLOCK_SelectIHOSC_CalHAO 30
4. TIMER/WDT Driver 32
4.1. Introduction 32

© 2021-2022HYCON Technology Corp

4.2. Type Definition 33
4.3. Functions 35
4.3.1. DrvWDT_Open 35
4.3.2. DrvWDT_CounterRead 35
4.3.3. DrvWDT_ClearWDT 36
4.3.4. DrvWDT_ResetEnable 36
4.3.5. DrvTMA_Open 37
4.3.6. DrvTMA_Close 38
4.3.7. DrvTMA_CounterRead 38
4.3.8. DrvTMA_ClearTMA 38
4.3.9. DrvTIMER_EnableInt 39
4.3.10. DrvTIMER_DisableInt 39
4.3.11. DrvTIMER_GetIntFlag 40
4.3.12. DrvTIMER_ClearIntFlag 40
4.3.13. DrvTMB_Open 41
4.3.14. DrvTMBC_Clk_Source 42
4.3.15. DrvTMBC_Clk_Disable 43
4.3.16. DrvTMB_ClearTMB 43
4.3.17. DrvTMB_CounterRead 44
4.3.18. DrvTMB_Close 44
4.3.19. DrvPWM0_Open 44
4.3.20. DrvPWM1_Open 45
4.3.21. DrvPWM_CountCondition 46
4.3.22. DrvPWM0_Close 47
4.3.23. DrvPWM1_Close 47
4.3.24. DrvCAPTURE1_Open 47
4.3.25. DrvCAPTURE2_Open 48
4.3.26. DrvCAPTURE1_Read 49
4.3.27. DrvCAPTURE2_Read 49
4.3.28. DrvCAPTURE_IPort 50
4.3.29. DrvTMB_TCI1Edge 50
4.3.30. DrvTMB_CPI1Input 51
4.3.31. DrvTMB2_Open 52
4.3.32. DrvTMB2_Close 53
4.3.33. DrvTMB2_Clk_Source 53
4.3.34. DrvTMB2_Clk_Disable 54
4.3.35. DrvTMB2_ClearTMB 54
4.3.36. DrvTMB2_CounterRead 55
4.3.37. DrvPWM2_Open 55

© 2021-2022HYCON Technology Corp

4.3.38. DrvPWM3_Open 56
4.3.39. DrvTMB2PWM_CountCondition 57
4.3.40. DrvPWM2_Close 57
4.3.41. DrvPWM3_Close 58
4.3.42. DrvTMB2_CPI3Input 58
4.3.43. DrvTMB2_TCI3Edge 59
5. GPIO Driver 60
5.1. Introduction 60
5.2. Type Definition 62
5.3. Functions 64
5.3.1. DrvGPIO_Open 64
5.3.2. DrvGPIO_Close 64
5.3.3. DrvGPIO_SetBit 65
5.3.4. DrvGPIO_ClrBit 66
5.3.5. DrvGPIO_GetBit 66
5.3.6. DrvGPIO_SetPortBits 67
5.3.7. DrvGPIO_ClrPortBits 67
5.3.8. DrvGPIO_GetPortBits 68
5.3.9. DrvGPIO_IntTrigger 68
5.3.10. DrvGPIO_ClearIntFlag 69
5.3.11. DrvGPIO_GetIntFlag 70
5.3.12. DrvGPIO_PortIDIF 70
5.3.13. DrvGPIO_LCDIOOpen 71
5.3.14. DrvGPIO_LCDIOClose 71
5.3.15. DrvGPIO_LCDIOSetPorts 72
5.3.16. DrvGPIO_LCDIOClrPorts 73
5.3.17. DrvGPIO_LCDIOSetBit 74
5.3.18. DrvGPIO_LCDIOClrBit 74
5.3.19. DrvGPIO_LCDIOGetPorts 75
5.3.20. DrvGPIO_LCDIOGetBit 76
5.3.21. DrvGPIO_EnableAnalogPin 77
5.3.22. DrvGPIO_PT1_EnableINPUT 77
5.3.23. DrvGPIO_PT1_DisableINPUT 78
5.3.24. DrvGPIO_PT1_EnablePullHigh 78
5.3.25. DrvGPIO_PT1_DisablePullHigh 79
5.3.26. DrvGPIO_PT1_EnableOUTPUT 79
5.3.27. DrvGPIO_PT1_DisableOUTPUT 80
5.3.28. DrvGPIO_PT1_EnableINT 80
5.3.29. DrvGPIO_PT1_DisableINT 81

© 2021-2022HYCON Technology Corp

5.3.30. DrvGPIO_PT1_IntTriggerPorts 81
5.3.31. DrvGPIO_PT1_IntTriggerBit 82
5.3.32. DrvGPIO_PT1_GetIntFlag 82
5.3.33. DrvGPIO_PT1_ClearIntFlag 83
5.3.34. DrvGPIO_PT1_GetPortBits 83
5.3.35. DrvGPIO_PT1_SetPortBits 84
5.3.36. DrvGPIO_PT1_ClrPortBits 84
5.3.37. DrvGPIO_PT2_EnableINPUT 85
5.3.38. DrvGPIO_PT2_DisableINPUT 85
5.3.39. DrvGPIO_PT2_EnablePullHigh 86
5.3.40. DrvGPIO_PT2_DisablePullHigh 86
5.3.41. DrvGPIO_PT2_EnableOUTPUT 87
5.3.42. DrvGPIO_PT2_DisableOUTPUT 87
5.3.43. DrvGPIO_PT2_EnableINT 88
5.3.44. DrvGPIO_PT2_DisableINT 88
5.3.45. DrvGPIO_PT2_IntTriggerPorts 89
5.3.46. DrvGPIO_PT2_IntTriggerBit 89
5.3.47. DrvGPIO_PT2_GetIntFlag 90
5.3.48. DrvGPIO_PT2_ClearIntFlag 90
5.3.49. DrvGPIO_PT2_GetPortBits 91
5.3.50. DrvGPIO_PT2_SetPortBits 91
5.3.51. DrvGPIO_PT2_ClrPortBits 92
5.3.52. DrvGPIO_PT3_EnableINPUT 92
5.3.53. DrvGPIO_PT3_DisableINPUT 93
5.3.54. DrvGPIO_PT3_EnablePullHigh 93
5.3.55. DrvGPIO_PT3_DisablePullHigh 94
5.3.56. DrvGPIO_PT3_EnableOUTPUT 94
5.3.57. DrvGPIO_PT3_DisableOUTPUT 95
5.3.58. DrvGPIO_PT3_EnableINT 95
5.3.59. DrvGPIO_PT3_DisableINT 96
5.3.60. DrvGPIO_PT3_IntTriggerPorts 96
5.3.61. DrvGPIO_PT3_IntTriggerBit 97
5.3.62. DrvGPIO_PT3_GetIntFlag 97
5.3.63. DrvGPIO_PT3_ClearIntFlag 98
5.3.64. DrvGPIO_PT3_GetPortBits 98
5.3.65. DrvGPIO_PT3_SetPortBits 99
5.3.66. DrvGPIO_PT3_ClrPortBits 99
5.3.67. DrvGPIO_PT6_EnableINPUT 100
5.3.68. DrvGPIO_PT6_DisableINPUT 100

© 2021-2022HYCON Technology Corp

5.3.69. DrvGPIO_PT6_EnableOUTPUT 101
5.3.70. DrvGPIO_PT6_DisableOUTPUT 101
5.3.71. DrvGPIO_PT6_GetPortBits 102
5.3.72. DrvGPIO_PT6_SetPortBits 102
5.3.73. DrvGPIO_PT6_ClrPortBits 103
5.3.74. DrvGPIO_PT7_EnableINPUT 103
5.3.75. DrvGPIO_PT7_DisableINPUT 104
5.3.76. DrvGPIO_PT7_EnableOUTPUT 104
5.3.77. DrvGPIO_PT7_DisableOUTPUT 105
5.3.78. DrvGPIO_PT7_GetPortBits 105
5.3.79. DrvGPIO_PT7_SetPortBits 106
5.3.80. DrvGPIO_PT7_ClrPortBits 106
5.3.81. DrvGPIO_PT8_EnableINPUT 107
5.3.82. DrvGPIO_PT8_DisableINPUT 107
5.3.83. DrvGPIO_PT8_EnableOUTPUT 108
5.3.84. DrvGPIO_PT8_DisableOUTPUT 108
5.3.85. DrvGPIO_PT8_GetPortBits 109
5.3.86. DrvGPIO_PT8_SetPortBits 109
5.3.87. DrvGPIO_PT8_ClrPortBits 109
5.3.88. DrvGPIO_PT9_EnableINPUT 110
5.3.89. DrvGPIO_PT9_DisableINPUT 110
5.3.90. DrvGPIO_PT9_EnableOUTPUT 111
5.3.91. DrvGPIO_PT9_DisableOUTPUT 111
5.3.92. DrvGPIO_PT9_GetPortBits 112
5.3.93. DrvGPIO_PT9_SetPortBits 112
5.3.94. DrvGPIO_PT9_ClrPortBits 113
5.3.95. DrvGPIO_PT10_EnableINPUT 113
5.3.96. DrvGPIO_PT10_DisableINPUT 114
5.3.97. DrvGPIO_PT10_EnableOUTPUT 114
5.3.98. DrvGPIO_PT10_DisableOUTPUT 115
5.3.99. DrvGPIO_PT10_GetPortBits 115
5.3.100. DrvGPIO_PT10_SetPortBits 116
5.3.101. DrvGPIO_PT10_ClrPortBits 116
5.3.102. DrvGPIO_PT13_EnableINPUT 117
5.3.103. DrvGPIO_PT13_DisableINPUT 117
5.3.104. DrvGPIO_PT13_EnableOUTPUT 118
5.3.105. DrvGPIO_PT13_DisableOUTPUT 118
5.3.106. DrvGPIO_PT13_GetPortBits 119
5.3.107. DrvGPIO_PT13_SetPortBits 119

© 2021-2022HYCON Technology Corp

5.3.108. DrvGPIO_PT13_ClrPortBits 120
6. ADC Driver 121
6.1. Introduction 121
6.2. Type Definition 122
6.3. Functions 124
6.3.1. DrvADC_PInputChannel 124
6.3.2. DrvADC_NInputChannel 124
6.3.3. DrvADC_SetADCInputChannel 125
6.3.4. DrvADC_InputSwitch 126
6.3.5. DrvADC_RefInputShort 126
6.3.6. DrvADC_SetPGA 127
6.3.7. DrvADC_ADGain 127
6.3.8. DrvADC_Gain 128
6.3.9. DrvADC_DCoffset 129
6.3.10. DrvADC_RefVoltage 130
6.3.11. DrvADC_FullRefRange 130
6.3.12. DrvADC_OSR 131
6.3.13. DrvADC_ACM 132
6.3.14. DrvADC_ClkEnable 132
6.3.15. DrvADC_ClkDisable 133
6.3.16. DrvADC_CombFilter 133
6.3.17. DrvADC_EnableInt 134
6.3.18. DrvADC_DisableInt 134
6.3.19. DrvADC_ReadIntFlag 135
6.3.20. DrvADC_ClearIntFlag 135
6.3.21. DrvADC_Enable 136
6.3.22. DrvADC_Disable 136
6.3.23. DrvADC_GetConversionData 136
7. SPI32 Driver 138
7.1. Introduction 138
7.2. Type Definition 138
E_DRVSPI_MODE 138
E_DRVSPI_TRANS_TYPE 138
E_DRVSPI_ENDIAN 139
E_DRVSPI_CS 139
7.3. Functions 140
7.3.1. DrvSPI32_Open 140
7.3.2. DrvSPI32_Close 141
7.3.3. DrvSPI32_IsBusy 141

© 2021-2022HYCON Technology Corp

7.3.4. DrvSPI32_CLKSource 142
7.3.5. DrvSPI32_IsRxBufferFull 142
7.3.6. DrvSPI32_IsTxBufferFull 143
7.3.7. DrvSPI32_EnableRxInt 143
7.3.8. DrvSPI32_EnableTxInt 144
7.3.9. DrvSPI32_DisableRxInt 144
7.3.10. DrvSPI32_DisableTxInt 145
7.3.11. DrvSPI32_GetRxIntFlag 145
7.3.12. DrvSPI32_GetTxIntFlag 146
7.3.13. DrvSPI32_ClrIntRxFlag 146
7.3.14. DrvSPI32_ClrIntTxFlag 146
7.3.15. DrvSPI32_Read 147
7.3.16. DrvSPI32_Write 147
7.3.17. DrvSPI32_Enable 148
7.3.18. DrvSPI32_BitLength 148
7.3.19. DrvSPI32_GetDCFlag 149
7.3.20. DrvSPI32_IsABFlag 149
7.3.21. DrvSPI32_IsOVFlag 150
7.3.22. DrvSPI32_IsRxFlag 150
7.3.23. DrvSPI32_SetEndian 151
7.3.24. DrvSPI32_SetCSO 151
7.3.25. DrvSPI32_DisableIO 152
7.3.26. DrvSPI32_EnableIO 152
8. UART Driver 154
8.1. Introduction 154
8.2. Type Definition 156
8.3. Functions 157
8.3.1. DrvUART_Open 157
8.3.2. DrvUART_Close 158
8.3.3. DrvUART_EnableInt 159
8.3.4. DrvUART_GetTxFlag 159
8.3.5. DrvUART_GetRxFlag 160
8.3.6. DrvUART_ClrTxFlag 160
8.3.7. DrvUART_ClrRxFlag 160
8.3.8. DrvUART_Read 161
8.3.9. DrvUART_ClrABDOVF 161
8.3.10. DrvUART_Write 162
8.3.11. DrvUART_EnableWakeUp 162
8.3.12. DrvUART_DisableWakeUp 163

© 2021-2022HYCON Technology Corp

8.3.13. DrvUART_GetPERR 163
8.3.14. DrvUART_GetFERR 164
8.3.15. DrvUART_GetOERR 164
8.3.16. DrvUART_GetABDOVF 164
8.3.17. DrvUART_Enable_AutoBaudrate 165
8.3.18. DrvUART_Disable_AutoBaudrate 165
8.3.19. DrvUART_CheckTRMT 166
8.3.20. DrvUART_ClkEnable 166
8.3.21. DrvUART_ClkDisable 167
8.3.22. DrvUART_Enable 167
8.3.23. DrvUART_ConfigIO 168
8.3.24. DrvUART_TRStatus 169
8.3.25. DrvUART_IntType 169
8.3.26. DrvUART_GetNERR 170
8.3.27. DrvUART_ClrPERR 170
8.3.28. DrvUART_ClrFERR 171
8.3.29. DrvUART_ClrOERR 171
8.3.30. DrvUART_ClrNERR 171
8.3.31. DrvUART2_Open 172
8.3.32. DrvUART2_Enable 173
8.3.33. DrvUART2_Close 174
8.3.34. DrvUART2_EnableInt 174
8.3.35. DrvUART2_IntType 175
8.3.36. DrvUART2_GetTxFlag 175
8.3.37. DrvUART2_GetRxFlag 176
8.3.38. DrvUART2_ClrTxFlag 176
8.3.39. DrvUART2_ClrRxFlag 177
8.3.40. DrvUART2_Read 177
8.3.41. DrvUART2_Write 178
8.3.42. DrvUART2_EnableWakeUp 178
8.3.43. DrvUART2_DisableWakeUp 179
8.3.44. DrvUART2_Enable_AutoBaudrate 179
8.3.45. DrvUART2_Disable_AutoBaudrate 179
8.3.46. DrvUART2_GetPERR 180
8.3.47. DrvUART2_GetFERR 180
8.3.48. DrvUART2_GetOERR 181
8.3.49. DrvUART2_GetNERR 181
8.3.50. DrvUART2_ClrPERR 182
8.3.51. DrvUART2_ClrFERR 182

© 2021-2022HYCON Technology Corp

8.3.52. DrvUART2_ClrOERR 183
8.3.53. DrvUART2_ClrNERR 183
8.3.54. DrvUART2_GetABDOVF 183
8.3.55. DrvUART2_ClrABDOVF 184
8.3.56. DrvUART2_TRStatus 184
8.3.57. DrvUART2_CheckTRMT 185
8.3.58. DrvUART2_ClkEnable 185
8.3.59. DrvUART2_ClkDisable 186
8.3.60. DrvUART2_ConfigIO 187
9. PMU Driver 188
9.1. Introduction 188
9.2. Type Definition 188
9.3. Functions 190
9.3.1. DrvPMU_VDD15Trim 190
9.3.2. DrvPMU_VDDA_Voltage 190
9.3.3. DrvPMU_VDDA_LDO_Ctrl 191
9.3.4. DrvPMU_BandgapEnable 191
9.3.5. DrvPMU_BandgapDisable 192
9.3.6. DrvPMU_REFO_Enable 192
9.3.7. DrvPMU_REFO_Disable 192
9.3.8. DrvPMU_AnalogGround 193
9.3.9. DrvPMU_LDO_LowPower 193
9.3.10. DrvPMU_GetLVDO 194
9.3.11. DrvPMU_EnableENLVD 194
9.3.12. DrvPMU_DisableENLVD 195
9.3.13. DrvPMU_SetLVDS 195
9.3.14. DrvPMU_SetLVDVS 196
9.3.15. DrvPMU_SetLVD12 196
9.3.16. DrvPMU_LVDIntTriMode 197
9.3.17. DrvPMU_EnableLVDInt 197
9.3.18. DrvPMU_DisableLVDInt 198
9.3.19. DrvPMU_ClrLVDIF 198
9.3.20. DrvPMU_EnableBOR2 199
9.3.21. DrvPMU_DisableBOR2 199
9.3.22. DrvPMU_BOR2_Mode 200
9.3.23. DrvPMU_DetectVol 200
9.3.24. DrvPMU_ReadBOR2Status 201
10. RTC Driver 202
10.1. Introduction 202

© 2021-2022HYCON Technology Corp

10.2. Type Definition 203
10.3. Functions 204
10.3.1. DrvRTC_SetFrequencyCompensation 204
10.3.2. DrvRTC_WriteEnable 204
10.3.3. DrvRTC_WriteDisable 205
10.3.4. DrvRTC_ClockSource 205
10.3.5. DrvRTC_AlarmEnable 206
10.3.6. DrvRTC_AlarmDisable 206
10.3.7. DrvRTC_PeriodicTimeEnable 207
10.3.8. DrvRTC_PeriodicTimeDisable 207
10.3.9. DrvRTC_Enable 208
10.3.10. DrvRTC_Disable 208
10.3.11. DrvRTC_HourFormat 209
10.3.12. DrvRTC_ReadState 209
10.3.13. DrvRTC_ClearState 210
10.3.14. DrvRTC_EnableInt 210
10.3.15. DrvRTC_DisableInt 211
10.3.16. DrvRTC_ReadIntFlag 211
10.3.17. DrvRTC_ClearIntFlag 212
10.3.18. DrvRTC_Write 212
10.3.19. DrvRTC_Read 213
11. I2C Driver 215
11.1. Introduction 215
11.2. Type Definition 216
11.3. Functions 217
11.3.1. DrvI2C_Open 217
11.3.2. DrvI2C_Close 217
11.3.3. DrvI2C_SlaveSet 218
11.3.4. DrvI2C_SlaveDisable 218
11.3.5. DrvI2C_SetIOPin 219
11.3.6. DrvI2C_WriteData 219
11.3.7. DrvI2C_Write3ByteData 220
11.3.8. DrvI2C_ReadData 220
11.3.9. DrvI2C_Ctrl 221
11.3.10. DrvI2C_EnableInt 221
11.3.11. DrvI2C_DisableInt 222
11.3.12. DrvI2C_ReadIntFlag 222
11.3.13. DrvI2C_ClearIntFlag 223
11.3.14. DrvI2C_ClearEIRQ 223

© 2021-2022HYCON Technology Corp

11.3.15. DrvI2C_ClearIRQ 224
11.3.16. DrvI2C_GetStatusFlag 224
11.3.17. DrvI2C_TimeOutEnable 226
11.3.18. DrvI2C_TimeOutDisable 227
11.3.19. DrvI2C_STSP 227
11.3.20. DrvI2C_MGetACK 228
11.3.21. DrvI2C_DisableIOPin 228
11.3.22. DrvI2C_Reset 229
12. LCD Driver 230
12.1. Introduction 230
12.2. Type Definition 231
12.3. Functions 232
12.3.1. DrvLCD_EnableCLK 232
12.3.2. DrvLCD_DisplayMode 232
12.3.3. DrvLCD_VLCDMode 233
12.3.4. DrvLCD_LcdDuty 233
12.3.5. DrvLCD_LCDBuffer 234
12.3.6. DrvLCD_LCDEnable 234
12.3.7. DrvLCD_VLCDEnable 235
12.3.8. DrvLCD_LCDBias 235
12.3.9. DrvLCD_IOMode 236
12.3.10. DrvLCD_WriteData 237
13. FLASH Read/Write Driver 239
13.1. Introduction 239
13.2. Functions 240
13.2.1. ISP_FUNC_ROMP->FlashOpEn 240
13.2.2. ISP_FUNC_ROMP->FlashOpDis 240
13.2.3. ISP_FUNC_ROMP->Burn_Word 241
13.2.4. ISP_FUNC_ROMP->BurnPage 241
13.2.5. ReadWord 242
13.2.6. ReadPage 242
13.2.7. ISP_FUNC_ROMP->SectorErase 243
13.2.8. ISP_FUNC_ROMP->CRC 244
13.2.9. ISP_FUNC_ROMP->fastBlank 244
13.2.10. The storage structure of Flash 245
14. Revision History 246
15. C Library Change List 247

© 2021-2022HYCON Technology Corp

Attention :

1、 HYCON Technology Corp. reserves the right to change the content of this datasheet without further notice. For most up-to-date information, please constantly visit our website: http://www.hycontek.com .

2、 HYCON Technology Corp. is not responsible for problems caused by figures or application circuits narrated herein whose related industrial properties belong to third parties.

3、 Specifications of any HYCON Technology Corp. products detailed or contained herein stipulate the performance, characteristics, and functions of the specified products in the independent state. We does not guarantee of the performance, characteristics, and functions of the specified products as placed in the customer's products or equipment. Constant and sufficient verification and evaluation is highly advised.

4、 Please note the operating conditions of input voltage, output voltage and load current and ensure the IC internal power consumption does not exceed that of package tolerance. HYCON Technology Corp. assumes no responsibility for equipment failures that resulted from using products at values that exceed, even momentarily, rated values listed in products specifications of HYCON products specified herein.

5、 Notwithstanding this product has built-in ESD protection circuit, please do not exert excessive static electricity to protection circuit.

6、 Products specified or contained herein cannot be employed in applications which require extremely high levels of reliability, such as device or equipment affecting the human body, health/medical equipments, security systems, or any apparatus installed in aircrafts and other vehicles.

7、 Despite the fact that HYCON Technology Corp. endeavors to enhance product quality as well as reliability in every possible way, failure or malfunction of semiconductor products may happen. Hence, users are strongly recommended to comply with safety design including redundancy and fire-precaution equipments to prevent any accidents and fires that may follow.

8、 Use of the information described herein for other purposes and/or reproduction or copying without the permission of HYCON Technology Corp. is strictly prohibited.

© 2021-2022HYCON Technology Corp

1. Overview

1.1. C Library Introduction

This document describes the HYCON ™ HY16F3910 series driver reference manual. System-level software developers can use the HYCON ™ HY16F3910 series driver to do the fast application software development, instead of using the register level programming, which can reduce the total development time significantly.

1.2. Relative Document

User can find the following documents in our website for other relative information.

http://www.hycontek.com/

2. SYS Driver

2.1. Introudction

The following functions are included in System Manager Section.

Item	Functions	Description
01	SYS_SleepFlagRead	Read Sleep Flag
02	SYS_SleepFlagClear	Clear Sleep Flag
03	SYS_WdogFlagRead	Read watch dog flag
04	SYS_WdogFlagClear	Clear watch dog flag
05	SYS_ResetFlagRead	Read reset flag of data
06	SYS_ResetFlagClear	Clear reset flag
07	SYS_BOR_FlagRead	Read BOR flag of data
08	SYS_BOR_FlagClear	Clear BOR flag
09	SYS_EnableGIE	Enable GIE and set priority level of interrupt
10	SYS_DisableGIE	Disable GIE
11	SYS_LowPower	Set the low power mode
12	SYS_INTPriority	Set the interrupt priority level of interrupt vector

© 2021-2022HYCON Technology Corp

2.2. Functions

2.2.1. SYS_SleepFlagRead

- **Prototype**

  unsigned int SYS_SleepFlagRead (void);

- **Description**

  Read Sleep Flag of data from registers 0x40104[3].

  Read the value of register 0x40104[3].

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  0 : Normal

  1 : Chip has entered Sleep Mode

- **Example**

```
/* Read Sleep Flag of data from registers 0x40104[3]. */
unsigned char temp_flag;     temp_flag=SYS_SleepFlagRead();
```

2.2.2. SYS_SleepFlagClear

- **Prototype**

  void SYS_SleepFlagClear(void);

- **Description**

  Clear Sleep Flag.

  Clear the register 0x40104[3].

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  None

- **Example**

```
/*Clear sleep flag. */
SYS_SleepFlagClear();    //set 0x40104[3]=0
```

2.2.3. SYS_WdogFlagRead

- **Prototype**

  unsigned int SYS_WdogFlagRead (void);

- **Description**

  Read watch dog flag of data from registers 0x40104[2].

  Read the value of register 0x40104[2].

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  0 : Normal

  1 : Watch dog has triggered

- **Example**

```
/*Read watch dog flag of data from registers r 0x40104[2].*/
unsigned char flag;    flag=SYS_WdogFlagRead();
```

2.2.4. SYS_WdogFlagClear

- **Prototype**

  void SYS_WdogFlagClear(void);

- **Description**

  Clear watch dog flag

  Clear the register 0x40104[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  None

- **Example**

```
/*Clear watch dog flag */
SYS_WdogFlagClear();    //0x40104[2]=0
```

2.2.5. SYS_ResetFlagRead

- **Prototype**

  unsigned int SYS_ResetFlagRead (void);

- **Description**

  Read reset flag of data from registers 0x40104[1].

© 2021-2022HYCON Technology Corp

Read the value of register 0x40104[1]

- **Parameters**

None

- **Include**

Peripheral_lib/System.h

- **Return Vaule**

0 : Normal

1 : The Reset Pin has reset

- **Example**

```
/*The Reset Pin has reset */
unsigned char flag;    flag=SYS_ResetFlagRead();
```

2.2.6. SYS_ResetFlagClear

- **Prototype**

```
void SYS_ResetFlagClear(void);
```

- **Description**

Clear reset flag

Clear the value of register 0x40104[1]

- **Parameters**

None

- **Include**

Peripheral_lib/System.h

- **Return Vaule**

None

- **Example**

```
/* Clear reset flag    */
SYS_ResetFlagClear();    //0x40104[1]=0
```

2.2.7. SYS_BOR_FlagRead

- **Prototype**

```
unsigned int SYS_BOR_FlagRead (void);
```

- **Description**

Read BOR flag of data from registers 0x40104[0].

Read the value of register 0x40104[0]

- **Parameters**

None

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  0 : Normal

  1 : BOR has triggered

- **Example**

```
/*Read BOR flag of data from registers 0x40104[0]. */
unsigned char flag;    flag=SYS_BOR_FlagRead();
```

2.2.8. SYS_BOR_FlagClear

- **Prototype**

```
void SYS_BOR_FlagClear(void);
```

- **Description**

  Clear BOR flag

  Clear the value of register 0x40104[0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  None

- **Example**

```
/* Clear BOR flag    */
SYS_BOR_FlagClear();    //0x40104[0]=0
```

2.2.9. SYS_EnableGIE

- **Prototype**

```
unsigned int SYS_EnableGIE (unsigned int uPriority,unsigned short intvector);
```

- **Description**

  Enable GIE and the corresponding interrupt vector, set the priority level of interrupt. The high level of priority will be responsed first. The priority of interrupt vector is set by SYS_INTPriority().

- **Parameters**

  uPriority [in] :

  Specify which priority level of interrupt can be responsed. It could be 0~4

  The priority level of the corresponding interrupt can be specified by SYS_INTPriority().

  0: No interrupts are allowed

  1: Only allows interrupts with the highest priority level.

© 2021-2022HYCON Technology Corp

2: Only allows interrupts with the highest and second highest priority level.

3: Only allows interrupts with the highest,second highest and lower priority level.

4: Only allows interrupts with the highest,second highest,lower and lowest priority level

intvector[in] :Select the interrupt vector[HW8:HW7:HW6:HW5:HW4:HW3:HW2:HW1:HW0]；It could be 0~0x1FF. Each bit corresponding to an interrupt vector:HW8~HW0

For exemple： invector=[HW8:HW7:HW6:HW5:HW4:HW3:HW2:HW1:HW0]

Only enable ：HW0/HW3/HW5, invectior=0x29（101001B）;

Enable all interrupt vector: invector=0x1FF（111111111B）;

Disable all interrupt vector: invector=0x00（000000B）

- **Return Vaule**
- **Include**

System.h

- **Return Vaule**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* Enable GIE and allows interrupts with priority 0, 1, 2,3, enableHW0~HW8. */
SYS_EnableGIE(4, 0x1FF);
```

2.2.10. SYS_DisableGIE

- **Prototype**

void SYS_DisableGIE (void);

- **Description**

Disable GIE

- **Parameters**

None

- **Include**

Peripheral_lib/System.h

- **Return Vaule**

None

- **Example**

```
/* Disable GIE. */
SYS_DisableGIE();
```

2.2.11. SYS_LowPower

- **Prototype**

unsigned char SYS_LowPower(unsigned char umode,unsigned char udisclk)

- **Description**

Set up and enable the low power mode.

Need to open any an interrupt vector before open low power mode and switch to a low frequency source

Set up the register 0x40104[4]、0x40300[18:16].

- **Parameters**

umode[in] : the input range is 0~2

0: sleep mode

1: idle mode

2: waite mode

udisclk[in] : the input range is 0~1

0 : Disable HAO(at idle mode)/Disable LPO(at sleep mode）

1 : Keep HAO(at idle mode）/Keep LPO(at sleep mode)

- **Include**

Peripheral_lib/System.h

- **Return Vaule**

0: Operation successful

1: Incorrect argument

- **Example**

```
/*Enable the sleep mode and disable LPO */
DrvGPIO_Open(E_PT1,0xFF,E_IO_IntEnable); // enable PT1 external interrupt vector
SYS_EnableGIE(4,0x1FF);    // Enable GIE(Global Interrupt)
DrvCLOCK_SelectMCUClock(0,0); // switch to a low frequency source
```

2.2.12. SYS_INTPriority

- **Prototype**

unsigned char SYS_INTPriority(unsigned short intvector,unsigned short upriority);

- **Description**

Specify priority level of the corresponding interrupt. Priority level is 0~3. 0 is the highest level

Note : Before use, must disable all interrupt to modify the interrupt priority level.

- **Parameters**

intvector[in] : the interrupt vector selection, input range of 0 to 9, respectively HW0 ~ HW9.

upriority [in] : set up and enable the priority level of interrupt vector，setting range is from 0 to 3

0: the highest level of priority level

1: the second highest level of priority level

2: the lower level of priority level.

3: the lowest level of priority level

When set the interrupt priority level for the same level, the order for the interrupt response：

HW0 > HW1 > HW2 > …> HW9

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/System.h

- **Return Vaule**

  0: Operation successful

  1: Incorrect argument

- **Example**

  /* set the priority level of interrupt vector 0 as 1 * /

  SYS_ INTPriority(0,1);

3. CLOCK Driver

3.1. Introduction

The following functions are included in Clock Manager Section.

Item	Functions	Description
01	DrvCLOCK_EnableHighOSC	Open high-speed oscillator
02	DrvCLOCK_CloseEHOSC	Turn off the external HSXT
03	DrvCLOCK_CloseIHOSC	Turn off the internal HSRC
04	DrvCLOCK_SelectIHOSC	Select HSRC mode
05	DrvCLOCK_EnableLowOSC	Open low-speed oscillator
06	DrvCLOCK_CloseELOSC	Turn off the external LSXT
07	DrvCLOCK_SelectMCUClock	Select the MCU Clock
08	DrvCLOCK_TrimHAO	Write to the HSRC Trim value
09	DrvCLOCK_CalibrateHAO	According to the factory calibration parameters of HAO to calibrate HAO
10	DrvCLOCK_SelectOHS_HS	Selecting External high-speed oscillator HSXT mode
11	DrvCLOCK_SelectIHOSC_CalHAO	Select HSRC mode and calibrate HAO

© 2021-2022HYCON Technology Corp

3.2. Type Definition

E_CLOCK_SOURCE

Enumeration Identifier	Value	Description
E_INTERNAL	0x0	Internal oscillator
E_EXTERNAL	0x1	External oscillator

E_MCUCK_SOURCE

Enumeration Identifier	Value	Description
E_HSCK	0x0	High speed oscillator
E_LSCK	0x1	Low speed oscillator

E_HAO_CLOCK

Enumeration Identifier	Value	Description
E_HAO_4M	0x1	Selesct the frequency of HAO 4.147MHZ
E_HAO_32M	0x3	Select the frequency of HAO 31.795MHZ

E_MCUCK_Prescale

Enumeration Identifier	Value	Description
MCUCKDIV2	0x0	MCU Clock/2
MCUCKDIV4	0x1	MCU Clock/4
MCUCKDIV8	0x2	MCU Clock/8
MCUCKDIV1	0x3	MCU Clock/1

E_TRIM_FREQUEN

Enumeration Identifier	Value	Description
TRIM_HAO4MHZ	0x1	Calibrate the frequency of HAO 4.147MHZ
TRIM_HAO32MHZ	0x2	Calibrate the frequency of HAO 31.795MHZ

© 2021-2022HYCON Technology Corp

3.3. Functions

3.3.1. DrvCLOCK_EnableHighOSC

- **Prototype**

unsigned int DrvCLOCK_EnableHighOSC(E_CLOCK_SOURCE uSource, unsigned int delay)

- **Description**

Open a high-speed oscillator, select from the external or internalSet the waiting time needed to stabilize the crystal

Configure the register 0x40300[5]=1 , 0x40300[1]=1 if the external OSC is selected as CPU clock source.

Configure the register 0x40300[5]=0，0x40300[0]=1 if the internal OSC is selected as CPU clock source.

- **Parameters**

uSource [in] :

0: Internal

1: External

delay[in] : Set the waiting time needed to stabilize the crystal. Input range：1~0xFFFFFFFF

Note that the current CPU cycles CPU_CLK, stabilization time of oscillator: t=(1/CPU_CLK)*4000*delay；

Refer to the current instruction cycle and the crystal frequency to start before set the parameter

The time needed to stabilize the EXT OSC :

4MHZ/8MHZ about 30ms

The time needed to stabilize the HAO :

4.147MHZ about 0.5ms

31.795MHZ about 0.1ms

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* Open external high-speed oscillator, current CPU_CK=10MHZ/2, Open external 4MHZ, Delay 40ms
=(1/10MHZ/2)*4000*50*/
DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
```

3.3.2. DrvCLOCK_CloseEHOSC

- **Prototype**

void DrvCLOCK_CloseEHOSC()

- **Description**

Turn off the external high-speed oscillator . Need to turn on a available clock source before turn off the

external high-speed oscillator if the external high-speed oscillator is the CPU clock source.

Configure the register 0x40300[1]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

None

- **Example**

/*Turn off the external high-speed oscillator*/

DrvCLOCK_EnableHighOSC(E_INTERNAL,50); //Open internal high-speed oscillator

DrvCLOCK_CloseEHOSC(); / /Turn off the external high-speed oscillator

3.3.3. DrvCLOCK_CloseIHOSC

- **Prototype**

void DrvCLOCK_CloseIHOSC()

- **Description**

Turn off the internal high-speed oscillator before switching the CPU clock source to available source

Configure the register 0x40300[0]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

None

- **Example**

/* Turn on the external high-speed oscillator and turn off the internal high-speed oscillator that is the CPU clock source */

DrvCLOCK_EnableHighOSC(E_EXTERNAL,50); // Open external high-speed oscillator

DrvCLOCK_CloseIHOSC(); // Close internal high-speed oscillator

3.3.4. DrvCLOCK_SelectIHOSC

- **Prototype**

unsigned int DrvCLOCK_SelectIHOSC(uMode)

- **Description**

Select high-speed internal oscillator mode

© 2021-2022HYCON Technology Corp

Configure the register 0x40300[4:3]

- **Parameters**

uMode [in] :

E_HAO_4M : 4.147MHz, 0x40300[4:3]=01b

E_HAO_32M : 31.795MHz, 0x40300[4:3]=11b

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

0: Operation successful

other: Incorrect argument

- **Example**

```
/* Select high-speed internal oscillator 4.147MHz mode*/
DrvCLOCK_SelectIHOSC(E_HAO_4M);
```

3.3.5. DrvCLOCK_EnableLowOSC

- **Prototype**

unsigned int DrvCLOCK_EnableLowOSC(E_CLOCK_SOURCE uSource，unint delay)

- **Description**

Open the low-speed oscillator; select the oscillator from an external or internal. Set the waiting time needed to stabilize the crystal

Configure the register 0x40300[6]=1, 0x40300[2]=1

- **Parameters**

uSource [in] :

0: Internal LSRC

1: External LSXT

delay[in] : Set the waiting time needed to stabilize the crystal. Need to refer to the current instruction cycle. Input range：0~0xFFFFFFFF

The time needed to stabilize the EXT OSC : 32768HZ about 1.3s

The time needed to stabilize the internal OSC : about 510us

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

0: Operation successful

other: Incorrect argument

- **Example**

```
/* Open the external low-speed oscillator, set the delay time 1.3s */
DrvCLOCK_EnableLowOSC(E_EXTERNAL,130000);
```

© 2021-2022HYCON Technology Corp

3.3.6. DrvCLOCK_CloseELOSC

- **Prototype**

  void DrvCLOCK_CloseELOSC()

- **Description**

  Turn off the external low-speed oscillator

  Configure the register 0x40300[2]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

  None

- **Example**

```
/* Turn on the internal low-speed oscillator and then turn off the external low-speed oscillator*/
DrvCLOCK_EnableLowOSC(E_INTERNAL,130000); //turn on internal low-speed oscillator
DrvCLOCK_CloseELOSC();    //close external low-speed oscillator
```

3.3.7. DrvCLOCK_SelectMCUClock

- **Prototype**

  unsigned int DrvCLOCK_SelectMCUClock(uSource,uDiv)

- **Description**

  Select the MCU Clock from HS_CK, or LS_CK and Pre-scale.

  Configure register 0x40308[0] / 0x40308[20:19]。

- **Parameters**

  uSource [in] :

  0 : HS_CK

  1 : LS_CK

  uDiv [in] :

  0 : ÷2

  1 : ÷4

  2 : ÷8

  3 : ÷1

- **Include**

  Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

  0: Operation successful

  other: Incorrect argument

© 2021-2022HYCON Technology Corp

- **Example**

```
/* Select the MCU Clock from HS_CK and Pre-scale 2 */
DrvCLOCK_SelectIHOSC(0,0);
```

3.3.8. DrvCLOCK_TrimHAO

- **Prototype**

unsigned int DrvCLOCK_TrimHAO(uTrim)

- **Description**

Write to the internal oscillator HAO Trim value

Configure the register 0x40304[7:0]

- **Parameters**

uTrim [in] : the internal oscillator Trim value, the input range is : 0~0xFF

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

0: Operation successful

other: Incorrect argument

- **Example**

```
/*Write 0x80 to the internal oscillator Trim value*/
DrvCLOCK_TrimHAO(0x80);
```

3.3.9. DrvCLOCK_CalibrateHAO

- **Prototype**

void DrvCLOCK_CalibrateHAO(short int uMHZ)

- **Description**

According to the factory calibration parameters of HAO to calibrate HAO, and need to corresponding to the selected HAO frequency. Configure the register 0x40304[7:0]

- **Parameters**

uMHZ [in] : the HAO frequency to calibrate

TRIM_HAO4MHZ：calibrate 4.147MHZ；

TRIM_HAO32MHZ：calibrate 31.795MHZ；

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

None

- **Example**

```
/* Calibrate internal OSC 4.147MHZ*/
```

© 2021-2022HYCON Technology Corp

DrvCLOCK_SelectIHOSC(TRIM_HAO4MHZ); //setting HAO=4.147MHZ;

DrvCLOCK_CalibrateHAO(TRIM_HAO4MHZ); //calibrate 4.147MHZ；

3.3.10. DrvCLOCK_SelectOHS_HS

- **Prototype**

unsigned int DrvCLOCK_SelectOHS_HS(unsigned int uMode)

- **Description**

Selecting External high-speed oscillator HSXT mode, the mode of HSXT can be more than 4MHz or less than 4MHz.

Configure the register 0x40300[7]

- **Parameters**

uMode [in] : Selecting External high-speed oscillator HSXT mode. The input range is : 0~1

0：HSXT<4MHz；1：HSXT>4MHz；

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

0: Operation successful

1: Incorrect argument

- **Example**

/*Select external high-speed oscillator (HSXT)>4MHZ */

DrvCLOCK_SelectOHS_HS(1); //Select HSXT > 4MHZ;

3.3.11. DrvCLOCK_SelectIHOSC_CalHAO

- **Prototype**

unsigned int DrvCLOCK_SelectIHOSC_CalHAO(uMode)

- **Description**

Select high-speed internal oscillator mode and according to the factory calibration parameters of HAO to calibrate HAO.

Configure the register0x40300[7]、0x40304[7:0]

- **Parameters**

uMode [in] :

E_HAO_4M : 4.147MHz, 0x40300[4:3]=01b

E_HAO_32M : 31.795MHz, 0x40300[4:3]=11b

- **Include**

Peripheral_lib/DrvCLOCK.h

- **Return Vaule**

0: Operation successful

© 2021-2022HYCON Technology Corp

other: Incorrect argument

- **Example**

/* Select high-speed internal oscillator 4.147MHz mode, and the error is less than 2% (after trim) */

DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);

© 2021-2022HYCON Technology Corp

4. TIMER/WDT Driver

4.1. Introduction

The following functions are included in Timer Manager Section.

Item	Functions	Description
01	DrvWDT_Open	Open WDT
02	DrvWDT_CounterRead	Read the current WDT counter
03	DrvWDT_ClearWDT	Watch dog timer clear
04	DrvWDT_ResetEnable	Enable Watch dog(WDT) Reset mode
05	DrvTMA_Open	Enable timer A
06	DrvTMA_Close	Close timer A
07	DrvTMA_CounterRead	Read the current TMA counter
08	DrvTMA_ClearTMA	Clear Timer A
09	DrvTIMER_EnableInt	Enable the specified timer interrupt.
10	DrvTIMER_DisableInt	Disable the specified timer interrupt
11	DrvTIMER_GetIntFlag	Get the interrupt flag status
12	DrvTIMER_ClearIntFlag	Clear the interrupt flag
13	DrvTMB_Open	Enable timer B
14	DrvTMBC_Clk_Source	Timer B,C clock source selection
15	DrvTMBC_Clk_Disable	Disable timer B,C clock
16	DrvTMB_ClearTMB	Clear Timer B
17	DrvTMB_CounterRead	Read the current TMB counter
18	DrvTMB_Close	Close timer B
19	DrvPWM0_Open	Enable PWM and PWM0 mode
20	DrvPWM1_Open	Enable PWM and PWM1 mode
21	DrvPWM_CountCondition	PWM count condition parameter
22	DrvPWM0_Close	PWM0 off
23	DrvPWM1_Close	PWM1 off
24	DrvCAPTURE1_Open	Enable Capture1
25	DrvCAPTURE2_Open	Enable Capture2
26	DrvCAPTURE1_Read	Read Capture1 counter
27	DrvCAPTURE2_Read	Read Capture2 counter
28	DrvCAPTURE_IPort	Select the capture input pin
29	DrvTMB_TCI1Edge	Select the trigger mode of TMB TCI1 input port
30	DrvTMB_CPI1Input	Set the input source in the mode of TMB CPI1
31	DrvTMB2_Open	Set TMB2
32	DrvTMB2_Close	Disable TMB2
33	DrvTMB2_Clk_Source	Set TMB2 clock source
34	DrvTMB2_Clk_Disable	Disable TMB2 cloock source
35	DrvTMB2_ClearTMB	Clear the counting register of Timer B2
36	DrvTMB2_CounterRead	Read the current TMB2 counter
37	DrvPWM2_Open	Enable PWM2 and operation mode selection
38	DrvPWM3_Open	Enable PWM3 and operation mode selection
39	DrvTMB2PWM_CountCondition	Set PWM2/PWM3 count condition parameter
40	DrvPWM2_Close	Disable PWM2
41	DrvPWM3_Close	Disable PWM3
42	DrvTMB2_CPI3Input	Set the input source in the mode of

© 2021-2022HYCON Technology Corp

		TMB2 CPI3
43	DrvTMB2_TCI3Edge	Select the trigger method of TMB2 TCI3 input source

4.2. Type Definition

E_WDT_MODE

Enumeration Identifier	Value	Description
E_IRQ	0x0	IRQ mode
E_RST	0x1	RESET mode

E_WDT_PRE_SCALER

Enumeration Identifier	Value	Description
E_PRE_SCALER_D2	0x0	WDT_CK / 1
E_PRE_SCALER_D8	0x1	WDT_CK / 4
E_PRE_SCALER_D32	0x2	WDT_CK / 16
E_PRE_SCALER_D128	0x3	WDT_CK / 64
E_PRE_SCALER_D512	0x4	WDT_CK / 256
E_PRE_SCALER_D2048	0x5	WDT_CK / 1024
E_PRE_SCALER_D8192	0x6	WDT_CK / 4096
E_PRE_SCALER_D32768	0x7	WDT_CK / 16384

E_TIMER_CHANNEL

Enumeration Identifier	Value	Description
E_TMA	0x0	Specify the timer channel – A
E_TMB	0x1	Specify the timer channel – B
E_TMC0	0x2	Specify the timer channel - C
E_TMC1	0x3	Specify the timer channel - C
E_WDT	0x4	Specify the timer channel - WDT
E_TMB2	0x5	Specify the timer channel – B2

E_TMB_MODE

Enumeration Identifier	Value	Description
E_TMB_MODE0	0x0	16-bit saw tooth waveform count up TBC0 for the maximum limit
E_TMB_MODE1	0x1	16-bit triangular waveform up and down the count range of 0 to TBC0
E_TMB_MODE2	0x2	The two independent 8-bit saw tooth type count, up to TBC0 bit 15-8 and bit 7-0 for the maximum limit
E_TMB_MODE3	0x3	The two 8-bit saw tooth type count, TBR[15:0] will be automatically added by 1, only after TBR[7:0] overflow

E_TRIGGER_SOURCE

Enumeration Identifier	Value	Description
E_TMB_NORMAL	0x0	Always Enable
E_TMB_CMP_HIGH	0x1	CMP high trigger
E_TMB_OP_HIGH	0x2	OP high trigger
E_TMB_GPIO_HIGH	0x3	GPIO high trigger

E_DRVTIMER_CLOCK_SOURCE

© 2021-2022HYCON Technology Corp

Enumeration Identifier	Value	Description
E_HS_CK	1	TMA clock source from HS_CK
E_HS_CB	2	TMA clock source from HS_CB
E_LS_CK	3	TMA clock source from HS_CK

E_CAPTURE_SOURCE

Enumeration Identifier	Value	Description
E_TMC_CMPO	0x0	Comparator output
E_TMC_OPOD	0x1	Rail-to-rail OP amp digital output
E_TMC_LSCK	0x2	Low speed clock source
E_TMC_TCI0	0x3	TC1 form I/O port
E_TMC_TCI1	0x0	TC2 form I/O port
E_TMC_ASTC0	0X1	Inupt source of TC2 is the same as TC1

© 2021-2022HYCON Technology Corp

4.3. Functions

4.3.1. DrvWDT_Open

- **Prototype**

  uint32_t DrvWDT_Open (E_WDT_MODE eMode , E_WDT_PRE_SCALER eWDTpreScaler)

- **Description**

  Enable WDT engine clock and set WDT time-out interval and set WDT mode.

  Configure the register 0x40108[2:0] / 0x40108[6] / 0x40108[4]=1

- **Parameters**

  eMode [in] : the operating mode of WDT

  0 : Timer mode

  1 : Reset mode

  eWDTpreScaler [in] : the prescaler of WDT clock source

  0 : WDT_CK / 1

  1 : WDT_CK / 4

  2 : WDT_CK / 16

  3 : WDT_CK / 64

  4 : WDT_CK / 256

  5 : WDT_CK / 1024

  6: WDT_CK / 4096

  7: WDT_CK / 16384

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  0: Operation successful

  1: WDT open fail

- **Example**

```
/* Set the WDT in IRQ mode and CLK / 16 */
DrvWDT_Open(E_IRQ , E_PRE_SCALER_D32);
```

4.3.2. DrvWDT_CounterRead

- **Prototype**

  uint32_t DrvWDT_CounterRead (void)

- **Description**

  Read the current WDT counter.

  Read the register 0x40108 [30:16]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

The return values of WDT counter.

- **Example**

/* Read the return values of WDT counter */

unsigned int data；data=DrvWDT_CounterRead();

4.3.3. DrvWDT_ClearWDT

- **Prototype**

void DrvWDT_ClearWDT (void)

- **Description**

Watch dog timer clear.

Configure the register 0x40108[5]=1, and 0x40108 [30:16] automatically becomes 0 after clear.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/* Watch dog timer clear. */

DrvWDT_ClearWDT();

4.3.4. DrvWDT_ResetEnable

- **Prototype**

void DrvWDT_ResetEnable(void)

- **Description**

Enable Watch dog(WDT) Reset mode .

Configure the register 0x40108[6]=1b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/* Enable Watch dog(WDT) Reset mode. */

© 2021-2022HYCON Technology Corp

```
DrvWDT_ResetEnable();
```

4.3.5. DrvTMA_Open

- **Prototype**

unsigned int DrvTMA_Open (eTMAOV, E_DRVTIMER_CLOCK_SOURCE uclk)

- **Description**

Enable timerA ,set counter value and clock source of TMA.

Configure the register 0x40C00[5]=1b, 0x40C00[3:0], 0x40308[3], 0x40308[2]

- **Parameters**

eTMAOV [in] : Specify timer A overflow condition.

0 : taclk/2

1 : taclk/4

2 : taclk/8

3 : taclk/16

4 : taclk/32

5 : taclk/64

6 : taclk/128

7 : taclk/256

8 : taclk/512

9 : taclk/1024

10 : taclk/2048

11 : taclk/4096

12 : taclk/8192

13 : taclk/16384

14 : taclk/32768

15: taclk/65536

uclk[in] : Specify timer A clock source.

1: HS_CK

2: HS_CB

3: LS_CK

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable timerA and set counter value is taclk/8. */
DrvTMA_Open(2, 1);
```

© 2021-2022HYCON Technology Corp

4.3.6. DrvTMA_Close

- **Prototype**

  void DrvTMA_Close (void)

- **Description**

  Close timerA

  Configure the register 0x40C00[5]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

  /* Disable timerA */

  DrvTMA_Close();

4.3.7. DrvTMA_CounterRead

- **Prototype**

  unsigned int DrvTMA_CounterRead (void)

- **Description**

  Read the current TMA counter.

  Read the register 0x40C00[15:0].

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  The return values of TMA counter.

- **Example**

  /* Read the current TMA counter */

  unsigned short tcounter; tcounter=DrvTMA_CounterRead();

4.3.8. DrvTMA_ClearTMA

- **Prototype**

  void DrvTMA_ClearTMA (void)

© 2021-2022HYCON Technology Corp

- **Description**

  Clear Timer A counter.

  Configure the register 0x40C00[4]=1, and 0x40C00[15:0] automatically becomes 0 after clear.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

  /* Clear Timer A. */

  DrvTMA_ClearTMA();

4.3.9. DrvTIMER_EnableInt

- **Prototype**

  unsigned int DrvTIMER_EnableInt (E_TIMER_CHANNEL ch)

- **Description**

  This function is used to enable the specified timer interrupt WDT/Timer A/Timer B/Timer B2/Timer C.

  Configure the corresponding bit of register 0x40004[20:16], 0x4001C[17] TimerB2 interrupt function=1

- **Parameters**

  ch [in] : timer interrupt source, the input range is 0~5

  0：TMA　　1：TMB　2：TMC C0

  3：TMC C1　4：WDT　5.TMB2

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  0: Operation successful

  Other : Invalid

- **Example**

  /* Enable Timer-A interrupt function */

  DrvTIMER_EnableInt(E_TMA);

4.3.10. DrvTIMER_DisableInt

- **Prototype**

  unsigned int DrvTIMER_DisableInt (E_TIMER_CHANNEL ch)

- **Description**

  This function is used to disable the specified timer interrupt WDT/Timer A/Timer B/Timer B2/Timer C.

Configure a corresponding bit of register 0x40004[20:16], 0x4001C[17] TimerB2 interrupt function=0

- **Parameters**

ch [in] : timer interrupt source, the input range is 0~5

0：TMA　　1：TMB　2：TMC C0

3：TMC C1　4：WDT　5:TMB2

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Invalid

- **Example**

```
/* Disable Timer-A interrupt function */
DrvTIMER_DisableInt(E_TMA);
```

4.3.11. DrvTIMER_GetIntFlag

- **Prototype**

unsigned int DrvTIMER_GetIntFlag (E_TIMER_CHANNEL ch)

- **Description**

Get the interrupt flag status from the specified timer channe WDT/Timer A/Timer B/Timer B2/Timer C.

Read a corresponding bit of register 0x40004[4:0] / 0x4001C[1] TimerB2

- **Parameters**

ch [in] : timer interrupt source, the input range is 0~5

0：TMA　　1：TMB　2：TMC C0

3：TMC C1　4：WDT　5：TMB

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: No interrupt

1: Interrupt occurred

- **Example**

```
/* Get the interrupt flag status from Timer-A */
DrvTIMER_GetIntFlag(E_TMA);
```

4.3.12. DrvTIMER_ClearIntFlag

- **Prototype**

unsigned int DrvTIMER_ClearIntFlag (E_TIMER_CHANNEL ch)

- **Description**

© 2021-2022HYCON Technology Corp

Clear the interrupt flag of the specified timer channel WDT/Timer A/Timer B/Timer B2/Timer C.

Clear a corresponding bit of register 0x40004[4:0]/0x4001C[1] TimerB2

- **Parameters**

ch [in] : timer interrupt source, the input range is 0~5

0：TMA　　1：TMB　2：TMC C0

3：TMC C1　4：WDT　5.TMB2

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Invalid

- **Example**

```
/* Clear Timer-A interrupt flag */
DrvTIMER_ClearIntFlag(E_TMA);
```

4.3.13. DrvTMB_Open

- **Prototype**

unsigned int DrvTMB_Open (E_TMB_MODE eTMBmode, E_TRIGGER_SOURCE eTriSource, eTMBOV)

- **Description**

Enable TMB and set TMB counter value and TMB mode and trigger source.

Support compare and capture and counting and timing functions.

Configure the register 0x40C0C[15:0], 0x40C04[3:0] / 0x40C04[5]=1b.

- **Parameters**

eTMBmode [in] : Specify timer B counting mode.

0: TMBR is in UP mode. In the UP mode, the TMBR is increase by 1 for every positive edge of TBCLK. If it is larger than TBC0, TMBR changes to 0 for next positive edge of TBCLK and TMBIF is change to 1. Then, TMBR starts to up count again.

1: TMBR is in UP/Down mode. In the UP mode, the TMBR is increase by 1 for every positive edge of TBCLK.

If it is equal to TBC0, TMBR changes to down mode and TMBR become to decrease by 1 for every positive edge of TBCLK. Until TMBR down count to 0, TMBIF changes to 1 and TMBR starts to up count again.

2: TMBR is in two 8-bit PWM mode. The TMBR is broke to two independent 8-bit UP counters: TMBR[15:8] and TMBR[7:0]. The TMBR[15:8] up limit is controlled by TBC0[15:8] and TMBR[7:0] up limit is controlled by TBC0[7:0]. Both of the TMBRs are increase by 1 for every positive edge of TBCLK. If TMBR[15:8] is equal to TBC0[15:8], then the next positive edge of TBCLK would make TMBR[15:8] to be 0. TMBIF still remains 0. If TMBR[7:0] is equal to TBC0[7:0], then the next positive edge of TBCLK would make TMBR[7:0]

to be 0. TMBIF changes to 1.

3: TMBR is in step increment mode. TMBR is break into two counters TMBR[15:8] and TMBR[7:0]. Both of them are in Up mode. However, the limit of TMBR[7:0] is controlled by TBC0[7:0]. The TMBR[7:0] is increase by 1 for every positive edge of TBCLK. If TMBR[7:0] is equal to TBC0[7:0], then it would change to 0 at next positive edge of TBCLK. Moreover, the TMBIF changes to 1 and TMBR[15:8] increases by 1.

eTriSource [in] : Specify TMB trigger source.

0: Always Enable

3:TMC output high trigger (CPI1)

eTMAOV [in] : Specify overflow condition. (0~0xffff)

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Invalid

- **Example**

```
/* Enable timerB mode0 and set overflow condition 0xffff and always trigger. */
DrvTMB_Open(E_TMB_MODE0, E_TMB_NORMAL,0xffff);
```

4.3.14. DrvTMBC_Clk_Source

- **Prototype**

unsigned int DrvTMBC_Clk_Source (E_DRVTIMER_CLOCK_SOURCE uclk, uPerScale)

- **Description**

Timer B,C clock source selection and clock divider selection.

Configure the register 0x40308[7:6], 0x40308[5:4]

- **Parameters**

uclk[in] : Specify timer B,C clock source, the input range is 1~3

1: HS_CK

2: HS_CB

3: LS_CK

uPerScale [in] : Specify timer B,C clock divider, , the input range is 0~3

0: ÷1

1: ÷2

2: ÷4

3: ÷8

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Invalid

- **Example**

/* Select the timer B clock from HS_CK, divider of 2. */

DrvTMB_Clk_Source(1,1);

4.3.15. DrvTMBC_Clk_Disable

- **Prototype**

viod DrvTMBC_Clk_Disable (viod)

- **Description**

Disable timer B,C clock.

Configure the register 0x40308[6]=0。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/* Disable timer B,C clock.*/

DrvTMBC_Clk_Disable();

4.3.16. DrvTMB_ClearTMB

- **Prototype**

void DrvTMB_ClearTMB (void)

- **Description**

Clear Timer B.

Configure the register 0x40C04[4]=1, and 0x40C08[15:0] automatically change to 0 after clear.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/* Clear Timer B counter */

DrvTMB_ClearTMB();

© 2021-2022HYCON Technology Corp

4.3.17. DrvTMB_CounterRead

- **Prototype**

  unsigned int DrvTMB_CounterRead (void)

- **Description**

  讀取定時計數器B(Timer B)的計數寄存器的值；

  讀取寄存器0x40C08[15:0]。

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  The return values of TMB counter.

- **Example**

  /* Read the current TMB counter */

  unsigned short TMB_counter; TMB_counter =DrvTMB_CounterRead();

4.3.18. DrvTMB_Close

- **Prototype**

  void DrvTMB_Close (void)

- **Description**

  Close timer B

  Configure the register 0x40C04[5]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

  /* Disable timerB */

  DrvTMB_Close();

4.3.19. DrvPWM0_Open

- **Prototype**

  unsigned int DrvPWM0_Open (uPWM_Mode , uInv, uOuputPin)

- **Description**

© 2021-2022HYCON Technology Corp

Enable PWM and PWM0 operation mode selection. Select IO port to output. PWM output inverse control

Configure the register 0x40C04[18:16] / 0x40C04[19]、0x40840[4:2] / 0x40840[0]=1b

- **Parameters**

uPWM_Mode [in] : PWM Operation mode selection

0: PWM A 1: PWM B

2: PWM C 3: PWM D

4 : PWM E 5 : PWM F

6 : PWM G 7 : PWM G

uInv[in] : PWM output inverse control.

0 : inverse

1 : Normal

uOuputPin[in] :PWM IO port selection

0 : Port 1.0 =PWMO0, Port 1.1 =PWMO1

1 : Port 1.4 =PWMO0, Port 1.5 =PWMO1

2 : Port 2.0 =PWMO0, Port 2.1 =PWMO1

3 : Port 2.4 =PWMO0, Port 2.5 =PWMO1

4 : Port 6.0 =PWMO0, Port 6.1 =PWMO1

5 : Port 7.4 =PWMO0, Port 7.5 =PWMO1

6 : Port 9.0 =PWMO0, Port 9.1 =PWMO1

7 : Port 8.0 =PWMO0, Port 8.1 =PWMO1

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable PWM, PWM0 working in PWMA mode, the reverse signal, PT1.0 Output. */
DrvPWM0_Open(0, 0, 0);
```

4.3.20. DrvPWM1_Open

- **Prototype**

unsigned int DrvPWM1_Open (uPWM_Mode , uInv, uOuputPin)

- **Description**

Enable PWM and PWM1 operation mode selection. Select IO port to output. PWM output inverse control

Set 0x40C04[23:20] / 0x40840[4:2] / 0x40840[1]=1b

- **Parameters**

uPWM_Mode [in] : PWM Operation mode selection

0: PWM A 1: PWM B

© 2021-2022HYCON Technology Corp

2: PWM C 3: PWM D

4 : PWM E 5 : PWM F

6 : PWM G 7 : PWM G

uInv[in] : PWM output inverse control.

0 : inverse

1 : Normal

uOuputPin[in] :PWM IO port selection

0 : Port 1.0 =PWMO0, Port 1.1 =PWMO1

1 : Port 1.4 =PWMO0, Port 1.5 =PWMO1

2 : Port 2.0 =PWMO0, Port 2.1 =PWMO1

3 : Port 2.4 =PWMO0, Port 2.5 =PWMO1

4 : Port 6.0 =PWMO0, Port 6.1 =PWMO1

5 : Port 7.4 =PWMO0, Port 7.5 =PWMO1

6 : Port 9.0 =PWMO0, Port 9.1 =PWMO1

7 : Port 8.0 =PWMO0, Port 8.1 =PWMO1

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/*Enable PWM, PWM1 working in PWMA mode, the reverse signal., PT1.1 output */
DrvPWM1_Open(0, 0, 0);
```

4.3.21. DrvPWM_CountCondition

- **Prototype**

void DrvPWM_CountCondition (uTBC2 , uTBC1)

- **Description**

PWM0/PWM1 count condition parameter (TBC2, TBC1) setting.

Configure the register 0x40C10[15:0](TBC1) / 0x40C10[31:16](TBC2)

- **Parameters**

uTBC1 [in] : PWM0 count condition, specify the TBC1 condition. (The range is 0~0xFFFF)

uTBC2 [in] : PWM1 count condition, specify the TBC2 condition. (The range is 0~0xFFFF)

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

```
/* Set TBC1, TBC2 value of 0x4000 */
DrvPWM_CountCondition(0x4000, 0x4000);
```

4.3.22. DrvPWM0_Close

- **Prototype**

  void DrvPWM0_Close (void)

- **Description**

  PWM0 off

  Configure the register 0x40840[0]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

```
/*PWM0 off */
DrvPWM0_Close();
```

4.3.23. DrvPWM1_Close

- **Prototype**

  void DrvPWM1_Close (void)

- **Description**

  PWM1 off

  Configure the register 0x40840[1]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

```
/*PWM1 off */
DrvPWM1_Close();
```

4.3.24. DrvCAPTURE1_Open

- **Prototype**

  unsigned int DrvCapture1_Open (CAPTURE_SOURCE uChannel , uDivider, uEdge)

- **Description**

Enable Capture1, Selected the input sources, pre-scale, the trigger source control.

Configure the register 0x40C14[21:20] / 0x40C14[19:16] / 0x40C14[1] / 0x40C14[0]=1.

- **Parameters**

uChannel [in] : Capture 1 input source selection, the input range is 0~3

0 : CMPO

1 : OPOD

2 : LS_CK

3 : TCI1

uDivider [in] : Input clock prescale, the input range is 0~15

0: ÷1	8: ÷256
1: ÷2	9: ÷512
2: ÷4	10: ÷1024
3: ÷8	11: ÷2048
4: ÷16	12: ÷4096
5: ÷32	13: ÷8192
6: ÷64	14: ÷16384
7: ÷128	15: ÷32768

uEdge [in] :

0 : Rising-edge trigger

1 : Falling-edge trigger

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/*Enable capture1, Choose TCI1 input , divided by 2048，rising-edge trigger */
DrvCapture1_Open(3, 11, 0);
```

4.3.25. DrvCAPTURE2_Open

- **Prototype**

unsigned int DrvCapture2_Open (CAPTURE_SOURCE uChannel, uEdge)

- **Description**

Enable Capture2, Selected the input sources, pre-scale, the trigger source control.

Configure the register 0x40C14[22] / 0x40C14[2] / 0x40C14[0]=1

- **Parameters**

uChannel [in] : Capture 2 input source selection.

0:TCI2 from GPIO

1: With the Capture1 the same trigger source

uEdge [in] :

0: Rising -edge trigger

1: Falling -edge trigger

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/*Enable capture2, choose with the Capture1 the same trigger source, rising -edge trigger. */

DrvCapture2_Open(1, 0);

4.3.26. DrvCAPTURE1_Read

- **Prototype**

unsigned int DrvCapture1_Read (void)

- **Description**

Read Capture1 counter.

Configure the register 0x40C18[15:0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

Capture the results TCR0(0~0xffff)

- **Example**

/*Read Capture1 counter */

unsigned short tcounter; tcounter=DrvCapture1_Read();

4.3.27. DrvCAPTURE2_Read

- **Prototype**

unsigned int DrvCapture2_Read (void)

- **Description**

Read Capture2 counter.

Configure the register 0x40C18[31:16]

- **Parameters**

© 2021-2022HYCON Technology Corp

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

Capture2 the results TCR1(0~0xffff)

- **Example**

/*Read Capture2 counter */

unsigned short tcounter; tcounter=DrvCapture2_Read();

4.3.28. DrvCAPTURE_IPort

- **Prototype**

unsigned int DrvCapture_Iport (uInputPin)

- **Description**

Select the capture input pin.

Configure the register 0x40840[7:5]

- **Parameters**

uInputPin[in] :

0 : Port 1.0 =TCI1, Port 1.1 =TCI2, Port 6.0 =TCI3

1 : Port 1.2 =TCI1, Port 1.3 =TCI2, Port 6.2 =TCI3

2 : Port 1.4 =TCI1, Port 1.5 =TCI2, Port 7.4 =TCI3

3 : Port 1.6 =TCI1, Port 1.7 =TCI2, Port 7.6 =TCI3

4 : Port 2.0 =TCI1, Port 2.1 =TCI2, Port 9.0 =TCI3

5 : Port 2.2 =TCI1, Port 2.3 =TCI2, Port 9.2 =TCI3

6 : Port 2.4 =TCI1, Port 2.5 =TCI2, Port 10.0 =TCI3

7 : Port 2.6 =TCI1, Port 2.7 =TCI2, Port 10.2 =TCI3

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Set capture input pin of Port 1.6=TCI1, Port1.7=TCI2 */

DrvCapture_Iport(3);

4.3.29. DrvTMB_TCI1Edge

- **Prototype**

unsigned char DrvTMB_TCI1Edge(unsigned int uedge)

© 2021-2022HYCON Technology Corp

- **Description**

  Select the TMB TCI1 input mode.

  Configure the register 0x40C14[23]

- **Parameters**

  uedge [in] : Select the trigger mode of TMB TCI1 input port

  0: level trigger

  1: rising edge trigger

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  0: Operation successful

  1: Incorrect argument

- **Example**

```
/* set rising edge trigger mode toTCI1   */
DrvTMB_CPI1Input(3)；  //select TCI1 as input source of CPI1 mode
DrvTMB_TCI1Edge(1);   //set as rising edge trigger for TCI1 IO；
```

4.3.30. DrvTMB_CPI1Input

- **Prototype**

  unsigned char DrvTMB_CPI1Input(unsigned int usource)

- **Description**

  Set the input source in the mode of TMB CPI1 .

  Configure the register 0x40C14[21:20]

- **Parameters**

  usource [in] : Set the input source in the mode of TMB CPI1

  0: CMPO comparator output

  1: R2R amplifier output

  2: LS_CK

  3: IO port

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  0: Operation successful

  1: Incorrect argument

- **Example**

```
/* Set TCI1 as the input source in the mode of TMB CPI1   */
DrvTMB_CPI1Input(3);
```

© 2021-2022HYCON Technology Corp

4.3.31. DrvTMB2_Open

● **Prototype**

unsigned int DrvTMB2_Open (E_TMB_MODE eTMBmode, E_TRIGGER_SOURCE eTriSource, eTMBOV)

● **Description**

Enable TMB2 and set TMB2 counter value and TMB2 mode and trigger source. Support compare and capture and counting and timing functions.

Configure the register 0x40C2C[15:0]、0x40C24[3:0] / 0x40C24[5]=1b

● **Parameters**

eTMBmode [in] : Specify timer B2 counting mode.

0: TMB2R is in UP mode. In the UP mode, the TMB2R is increase by 1 for every positive edge of TB2CLK. If it is larger than TB2C0, TMB2R changes to 0 for next positive edge of TB2CLK and TMB2IF is change to 1. Then, TMB2R starts to up count again.

1: TMB2R is in UP/Down mode. In the UP mode, the TMB2R is increase by 1 for every positive edge of TB2CLK.

If it is equal to TB2C0, TMB2R changes to down mode and TMB2R become to decrease by 1 for every positive edge of TB2CLK. Until TMB2R down count to 0, TMB2IF changes to 1 and TMB2R starts to up count again.

2: TMB2R is in two 8-bit PWM mode. The TMB2R is broke to two independent 8-bit UP counters: TMB2R [15:8] and TMB2R [7:0]. The TMB2R [15:8] up limit is controlled by TB2C0[15:8] and TMB2R [7:0] up limit is controlled by TB2C0[7:0]. Both of the TMB2Rs are increase by 1 for every positive edge of TB2CLK. If TMB2R [15:8] is equal to TB2C0[15:8], then the next positive edge of TB2CLK would make TMB2R [15:8] to be 0. TMB2IF still remains 0. If TMB2R [7:0] is equal to TB2C0[7:0], then the next positive edge of TB2CLK would make TMB2R [7:0] to be 0. TMB2IF changes to 1.

3: TMB2R is in step increment mode. TMB2R is break into two counters TMB2R[15:8] and TMB2R[7:0]. Both of them are in Up mode. However, the limit of TMB2R [7:0] is controlled by TB2C0[7:0]. The TMB2R [7:0] is increase by 1 for every positive edge of TB2CLK. If TMB2R [7:0] is equal to TB2C0[7:0], then it would change

to 0 at next positive edge of TB2CLK. Moreover, the TMB2IF changes to 1 and TMB2R [15:8] increases by 1.

eTriSource [in] : Specify TMB2 trigger source.

0: Always Enable

3:TMC output high trigger (CPI1)

eTMAOV [in] : Specify overflow condition. (0~0xffff)

● **Include**

Peripheral_lib/DrvTIMER.h

● **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/*Enable timerB2 mode0 and set overflow condition 0xffff and always trigger. */

DrvTMB2_Open(E_TMB_MODE0, E_TMB_NORMAL, 0xffff);

4.3.32. DrvTMB2_Close

- **Prototype**

void DrvTMB2_Close (void)

- **Description**

Close timer B2

Configure the register 0x40C24[5]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/ *Disable timerB2 */

DrvTMB2_Close();

4.3.33. DrvTMB2_Clk_Source

- **Prototype**

unsigned int DrvTMB2_Clk_Source (E_DRVTIMER_CLOCK_SOURCE uclk, uPerScale)

- **Description**

Timer B2 clock source selection and clock divider selection.

Configure the register 0x40314[7:4]

- **Parameters**

uclk[in] : Specify timer B2 clock source.

1: HS_CK

2: HS_CB

3: LS_CK

uPerScale [in] : Specify timer B2 clock divider.

0: ÷1 1: ÷2

2: ÷4 3: ÷8

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

© 2021-2022HYCON Technology Corp

Other : Incorrect argument

- **Example**

/* Select the timer B2 clock from HS_CK, divider of 2. */

DrvTMB2_Clk_Source(1,1);

4.3.34. DrvTMB2_Clk_Disable

- **Prototype**

viod DrvTMB2_Clk_Disable (viod)

- **Description**

Disable timer B2 clock.

Configure the register 0x40314[6]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/*Disable timer B2 clock. */

DrvTMB2_Clk_Disable();

4.3.35. DrvTMB2_ClearTMB

- **Prototype**

void DrvTMB2_ClearTMB (void)

- **Description**

Clear the counting register of Timer B2.

Configure the register 0x40C24[4]=1, and 0x40C28[15:0] automatically change to 0 after clear.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

None

- **Example**

/*Clear Timer B2 counter. */

DrvTMB2_ClearTMB();

© 2021-2022HYCON Technology Corp

4.3.36. DrvTMB2_CounterRead

- **Prototype**

  unsigned int DrvTMB2_CounterRead(void)

- **Description**

  Read the current TMB2 counter.

  Read the register 0x40C28[15:0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  The return values of TMB2 counter.

- **Example**

  /*Read the current TMB2 counter. */

  unsigned short tcounter; tcounter=DrvTMB2_CounterRead();

4.3.37. DrvPWM2_Open

- **Prototype**

  unsigned int DrvPWM2_Open (uPWM_Mode , uInv, uOuputPin)

- **Description**

  Enable PWM2 and PWM2 operation mode selection. Select IO port to output. PWM2 output inverse control

  Configure the register 0x40C24[18:16] / 0x40C24[19]、0x40848[4:2] / 0x40848[0]

- **Parameters**

  uPWM_Mode [in] : PWM2 Operation mode selection

  0: PWM A 1: PWM B

  2: PWM C 3: PWM D

  4 : PWM E 5 : PWM F

  6 : PWM G 7 : PWM G

  uInv[in] : PWM2 output inverse control.

  0 : inverse

  1 : Normal

  uOuputPin[in] : PWM2 IO port selection

  0 : Port 1.2 =PWMO2, Port 1.3 =PWMO3

  1 : Port 1.6 =PWMO2, Port 1.7 =PWMO3

  2 : Port 2.2 =PWMO2, Port 2.3 =PWMO3

  3 : Port 2.6 =PWMO2, Port 2.7 =PWMO3

  4 : Port 6.2 =PWMO2, Port 6.3 =PWMO3

© 2021-2022HYCON Technology Corp

5 : Port 7.6 =PWMO2, Port 7.7 =PWMO3

6 : Port 9.2 =PWMO2, Port 9.3 =PWMO3

7 : Port 8.2 =PWMO2, Port 8.3 =PWMO3

- **Include**

Peripheral_lib/DrvTIMER.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/*Enable PWM2, PWM2 working in PWMA mode, the reverse signal, PT1.2 output */

DrvPWM2_Open(0, 0, 0);

4.3.38. DrvPWM3_Open

- **Prototype**

unsigned int DrvPWM3_Open (uPWM_Mode , uInv, uOuputPin)

- **Description**

Enable PWM3 and PWM3 operation mode selection. Select IO port to output. PWM output inverse control

Configure the register 0x40C24[23:20], 0x40848[4:1].

- **Parameters**

uPWM_Mode [in] : PWM3 Operation mode selection

0: PWM A 1: PWM B

2: PWM C 3: PWM D

4 : PWM E 5 : PWM F

6 : PWM G 7 : PWM G

uInv[in] : PWM3 output inverse control.

0 : inverse

1 : Normal

uOuputPin[in] : PWM3 IO port selection

0 : Port 1.2 =PWMO2, Port 1.3 =PWMO3

1 : Port 1.6 =PWMO2, Port 1.7 =PWMO3

2 : Port 2.2 =PWMO2, Port 2.3 =PWMO3

3 : Port 2.6 =PWMO2, Port 2.7 =PWMO3

4 : Port 6.2 =PWMO2, Port 6.3 =PWMO3

5 : Port 7.6 =PWMO2, Port 7.7 =PWMO3

6 : Port 9.2 =PWMO2, Port 9.3 =PWMO3

7 : Port 8.2 =PWMO2, Port 8.3 =PWMO3

- **Include**

Peripheral_lib/DrvTIMER.h

© 2021-2022HYCON Technology Corp

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable PWM3, PWM3 working in PWMA mode, the reverse signal, PT1.3 */
DrvPWM3_Open(0, 0, 0);
```

4.3.39. DrvTMB2PWM_CountCondition

- **Prototype**

  void DrvTMB2PWM_CountCondition (uTBC2 , uTBC1)

- **Description**

  PWM2/PWM3 count condition parameter (TBC2, TBC1) setting.

  Configure the register 0x40C30[15:0](TBC1) / 0x40C30[15:0](TBC2)

- **Parameters**

  uTBC1 [in] : PWM2 count condition, specify the TBC1 condition. (The range is 0~0xFFFF)

  uTBC2 [in] : PWM3 count condition, specify the TBC2 condition. (The range is 0~0xFFFF)

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

```
/*Set TBC1, TBC2 value of 0x4000    */
DrvTMB2PWM_CountCondition(0x4000,0x4000);
```

4.3.40. DrvPWM2_Close

- **Prototype**

  void DrvPWM2_Close (void)

- **Description**

  PWM2 off

  Configure the register 0x40848[0]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

```
/*PWM2 off */
DrvPWM2_Close();
```

4.3.41. DrvPWM3_Close

- **Prototype**

  void DrvPWM3_Close (void)

- **Description**

  PWM3 off

  Configure the register 0x40848[1]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  None

- **Example**

```
/*PWM3 off */
DrvPWM3_Close();
```

4.3.42. DrvTMB2_CPI3Input

- **Prototype**

  unsigned char DrvTMB2_CPI3Input(unsigned int usource)

- **Description**

  Set the input source in the mode of TMB2 CPI3 .

  Configure the register 0x40C34[21:20]

- **Parameters**

  usource [in] : Set the input source in the mode of TMB2 CPI3

  0: CMPO comparator output

  1: R2R amplifier output

  2: LS_CK

  3: IO port

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  0: Operation successful

  1: Incorrect argument

- **Example `**

```
/*Set TCI3 as the input source in the mode of TMB2 CPI3 */
DrvTMB2_CPI3Input(3);
```

4.3.43. DrvTMB2_TCI3Edge

- **Prototype**

  unsigned char DrvTMB2_TCI3Edge(unsigned int uedge)

- **Description**

  Select the trigger method of TMB2 TCI3 input source.

  Configure the register 0x40C34[23]

- **Parameters**

  uedge [in] : Select the TMB2 TCI3 input mode

  0: level trigger

  1: rising edge trigger

- **Include**

  Peripheral_lib/DrvTIMER.h

- **Return Vaule**

  0: Operation successful

  1: Incorrect argument

- **Example**

```
/* set rising edge trigger mode toTCI1   */
DrvTMB2_CPI3Input(3)；  //select TCI1 as input source of CPI3 mode
DrvTMB2_TCI3Edge(1);   //set as rising edge trigger for TCI3 IO；
```

© 2021-2022HYCON Technology Corp

5. GPIO Driver

5.1. Introduction

The following functions are included in GPIO Manager Section.

Item	Functions	Description
01	DrvGPIO_Open	Set the GPIO operation mode
02	DrvGPIO_Close	Close the GPIO operation mode
03	DrvGPIO_SetBit	Set the specified GPIO pin to 1
04	DrvGPIO_ClrBit	Set the specified GPIO pin to 0.
05	DrvGPIO_GetBit	Get the pin value
06	DrvGPIO_SetPortBits	Set the output port value
07	DrvGPIO_ClrPortBits	Clear the output port value
08	DrvGPIO_GetPortBits	Get the input port value
09	DrvGPIO_IntTrigger	Set the specified interrupt pin mode
10	DrvGPIO_ClearIntFlag	Clear external interrupt flag
11	DrvGPIO_GetIntFlag	Get the interrupt flag
12	DrvGPIO_PortIDIF	Read the PT1~PT3 condition flag of interrupt trigger
13	DrvGPIO_LCDIOOpen	Set the GPIO(PT6~13) operation mode
14	DrvGPIO_LCDIOClose	Close the GPIO(PT6~13) operation mode
15	DrvGPIO_LCDIOSetPorts	Set the specified GPIO(PT6~13) pin to 1
16	DrvGPIO_LCDIOClrPorts	Set the specified GPIO(PT6~13) pin to 0.
17	DrvGPIO_LCDIOSetBit	Set one specified GPIO(PT6~13) pin to 1
18	DrvGPIO_LCDIOClrBit	Set one specified GPIO(PT6~13) pin to 0.
19	DrvGPIO_LCDIOGetPorts	Get the input port value from the GPIO(PT6~PT13) port
20	DrvGPIO_LCDIOGetBit	Get one input port value from the GPIO(PT6~PT13) port
21	DrvGPIO_EnableAnalogPin	Disabled the GPIO digital mode.Enable the GPIO analog mode
22	DrvGPIO_PT1_EnableINPUT	Enable the input mode for specified pin of PT1
23	DrvGPIO_PT1_DisableINPUT	Disable the input mode for specified pin of PT1
24	DrvGPIO_PT1_EnablePullHigh	Enable the pull up mode for specified pin of PT1
25	DrvGPIO_PT1_DisablePullHigh	Disable the pull up mode for specified pin of PT1
26	DrvGPIO_PT1_EnableOUTPUT	Enable the output mode for specified pin of PT1
27	DrvGPIO_PT1_DisableOUTPUT	Disable the output mode for specified pin of PT1
28	DrvGPIO_PT1_EnableINT	Enable the external interrupt for the specified pin of PT1
29	DrvGPIO_PT1_DisableINT	Disable the external interrupt for the specified pin of PT1
30	DrvGPIO_PT1_IntTriggerPorts	Configure the external interrupt trigger method for PT1
31	DrvGPIO_PT1_IntTriggerBit	Configure the external interrupt trigger method for the specified pin of PT1
32	DrvGPIO_PT1_GetIntFlag	Clear the interrupt flag of the specified pin
33	DrvGPIO_PT1_ClearIntFlag	Get the interrupt flag of the specified pin
34	DrvGPIO_PT1_GetPortBits	Get the input port value from the specified pin

© 2021-2022HYCON Technology Corp

35	DrvGPIO_PT1_SetPortBits	Set the output port value of the specified pin
36	DrvGPIO_PT1_ClrPortBits	Clear the output port value of the specified pin
37	DrvGPIO_PT2_EnableINPUT	Enable the input mode of the specified pin
38	DrvGPIO_PT2_DisableINPUT	Disable the input mode of the specified pin
39	DrvGPIO_PT2_EnablePullHigh	Enable the pull up of the specified pin
40	DrvGPIO_PT2_DisablePullHigh	Disable the pull up of the specified pin
41	DrvGPIO_PT2_EnableOUTPUT	Enable the output mode of the specified pin
42	DrvGPIO_PT2_DisableOUTPUT	Disable the output mode of the specified pin
43	DrvGPIO_PT2_EnableINT	Enable the external interrupt function of the specified pin
44	DrvGPIO_PT2_DisableINT	Disable the external interrupt function of the specified pin
45	DrvGPIO_PT2_IntTriggerPorts	Configure the external interrupt trigger method for PT2
46	DrvGPIO_PT2_IntTriggerBit	Configure the external interrupt trigger method for the specified pin of PT2
47	DrvGPIO_PT2_GetIntFlag	Clear the interrupt flag of the specified pin
48	DrvGPIO_PT2_ClearIntFlag	Get the interrupt flag of the specified pin
49	DrvGPIO_PT2_GetPortBits	Get the input port value from the specified pin
50	DrvGPIO_PT2_SetPortBits	Set the output port value of the specified pin
51	DrvGPIO_PT2_ClrPortBits	Clear the output port value of the specified pin
52	DrvGPIO_PT3_EnableINPUT	Enable the input mode of the specified pin
53	DrvGPIO_PT3_DisableINPUT	Disable the input mode of the specified pin
54	DrvGPIO_PT3_EnablePullHigh	Enable the pull up of the specified pin
55	DrvGPIO_PT3_DisablePullHigh	Disable the pull up of the specified pin
56	DrvGPIO_PT3_EnableOUTPUT	Enable the output mode of the specified pin
57	DrvGPIO_PT3_DisableOUTPUT	Disable the output mode of the specified pin
58	DrvGPIO_PT3_EnableINT	Enable the external interrupt function of the specified pin
59	DrvGPIO_PT3_DisableINT	Disable the external interrupt function of the specified pin
60	DrvGPIO_PT3_IntTriggerPorts	Configure the external interrupt trigger method for PT3
61	DrvGPIO_PT3_IntTriggerBit	Configure the external interrupt trigger method for the specified pin of PT3
62	DrvGPIO_PT3_GetIntFlag	Clear the interrupt flag of the specified pin
63	DrvGPIO_PT3_ClearIntFlag	Get the interrupt flag of the specified pin
64	DrvGPIO_PT3_GetPortBits	Get the input port value from the specified pin
65	DrvGPIO_PT3_SetPortBits	Set the output port value of the specified pin
66	DrvGPIO_PT3_ClrPortBits	Clear the output port value of the specified pin
67	DrvGPIO_PT6_EnableINPUT	Enable the input mode of the specified pin
68	DrvGPIO_PT6_DisableINPUT	Disable the input mode of the specified pin
69	DrvGPIO_PT6_EnableOUTPUT	Enable the output mode of the specified pin
70	DrvGPIO_PT6_DisableOUTPUT	Disable the output mode of the specified pin
71	DrvGPIO_PT6_GetPortBits	Get the input port value from the specified pin
72	DrvGPIO_PT6_SetPortBits	Set the output port value of the specified pin
73	DrvGPIO_PT6_ClrPortBits	Clear the output port value of the specified pin
74	DrvGPIO_PT7_EnableINPUT	Enable the input mode of the specified pin
75	DrvGPIO_PT7_DisableINPUT	Disable the input mode of the specified pin
76	DrvGPIO_PT7_EnableOUTPUT	Enable the output mode of the specified pin
77	DrvGPIO_PT7_DisableOUTPUT	Disable the output mode of the specified pin
78	DrvGPIO_PT7_GetPortBits	Get the input port value from the specified pin
79	DrvGPIO_PT7_SetPortBits	Set the output port value of the specified pin
80	DrvGPIO_PT7_ClrPortBits	Clear the output port value of the specified pin
81	DrvGPIO_PT8_EnableINPUT	Enable the input mode of the specified pin
82	DrvGPIO_PT8_DisableINPUT	Disable the input mode of the specified pin
83	DrvGPIO_PT8_EnableOUTPUT	Enable the output mode of the specified pin
84	DrvGPIO_PT8_DisableOUTPUT	Disable the output mode of the specified pin
85	DrvGPIO_PT8_GetPortBits	Get the input port value from the specified pin

© 2021-2022HYCON Technology Corp

86	DrvGPIO_PT8_SetPortBits	Set the output port value of the specified pin
87	DrvGPIO_PT8_ClrPortBits	Clear the output port value of the specified pin
88	DrvGPIO_PT9_EnableINPUT	Enable the input mode of the specified pin
89	DrvGPIO_PT9_DisableINPUT	Disable the input mode of the specified pin
90	DrvGPIO_PT9_EnableOUTPUT	Enable the output mode of the specified pin
91	DrvGPIO_PT9_DisableOUTPUT	Disable the output mode of the specified pin
92	DrvGPIO_PT9_GetPortBits	Get the input port value from the specified pin
93	DrvGPIO_PT9_SetPortBits	Set the output port value of the specified pin
94	DrvGPIO_PT9_ClrPortBits	Clear the output port value of the specified pin
95	DrvGPIO_PT10_EnableINPUT	Enable the input mode of the specified pin
96	DrvGPIO_PT10_DisableINPUT	Disable the input mode of the specified pin
97	DrvGPIO_PT10_EnableOUTPUT	Enable the output mode of the specified pin
98	DrvGPIO_PT10_DisableOUTPUT	Disable the output mode of the specified pin
99	DrvGPIO_PT10_GetPortBits	Get the input port value from the specified pin
100	DrvGPIO_PT10_SetPortBits	Set the output port value of the specified pin
101	DrvGPIO_PT10_ClrPortBits	Clear the output port value of the specified pin
102	DrvGPIO_PT13_EnableINPUT	Enable the input mode of the specified pin
103	DrvGPIO_PT13_DisableINPUT	Disable the input mode of the specified pin
104	DrvGPIO_PT13_EnableOUTPUT	Enable the output mode of the specified pin
105	DrvGPIO_PT13_DisableOUTPUT	Disable the output mode of the specified pin
106	DrvGPIO_PT13_GetPortBits	Get the input port value from the specified pin
107	DrvGPIO_PT13_SetPortBits	Set the output port value of the specified pin
108	DrvGPIO_PT13_ClrPortBits	Clear the output port value of the specified pin

5.2. Type Definition

E_DRVGPIO_PORT

Enumeration Identifier	Value	Description
E_PT1	1	Define GPIO Port 0
E_PT2	2	Define GPIO Port 1
E_PT3	3	Define GPIO Port 2

E_DRVGPIO_LCDIO

Enumeration Identifier	Value	Description
E_PT6	0	Define GPIO Port 6
E_PT7	1	Define GPIO Port 7
E_PT8	2	Define GPIO Port 8
E_PT9	3	Define GPIO Port 9
E_PT10	4	Define GPIO Port 10
E_PT13	7	Define GPIO Port 13

E_DRVGPIO_IO

Enumeration Identifier	Value	Description
E_IO_INPIT	0	Set GPIO as Input mode
E_IO_OUTPUT	1	Set GPIO as Output mode
E_IO_PullHigh	2	Pull High Enable
E_IO_ IntEnable	3	Interrupt Enable

© 2021-2022HYCON Technology Corp

E_DRVGPIO_IntTriMethod

Enumeration Identifier	Value	Description
E_DisableGPIOInt	0	Disable GPIO Interrupt
E_P_Edge	1	Positive Edge
E_N_Edge	2	Negative Edge
E_Chang_Level	3	Chang Level
E_LLTri	4	Level Low Trigger
E_LHTri	5	Level High Trigger
E_LLTri	6	Level Low Trigger
E_LHTri	7	Level High Trigger

© 2021-2022HYCON Technology Corp

5.3. Functions

5.3.1. DrvGPIO_Open

- **Prototype**

int32_t DrvGPIO_Open (E_DRVGPIO_PORT port, int32_t i32Bit, E_DRVGPIO_IO mode)

- **Description**

Set the specified GPIO(PT1~PT3) pin to the specified GPIO operation mode.

Configure the register

PT1 : 0x40800[23:16] / 0x40800[7:0] / 0x40804[23:16] / 0x40010[23:16]

PT2 : 0x40810[23:16] / 0x40810[7:0] / 0x40814[23:16] / 0x40014[23:16]

PT3 : 0x40820[23:16] / 0x40820[7:0] / 0x40824[23:16] / 0x40010[23:16]

- **Parameters**

port [in] : specify GPIO port, the effectively input range is 1~3

1：PT1　2：PT2　3：PT3.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be set if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

mode [in] : set the operation mode of the specified GPIO pin

0: Enable input mode　　1: Enable output mode

2: Enable pull up internally　　3: Enable external interrupt

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* configure PT2.0 to GPIO output mode and PT2.1 to GPIO input mode*/
DrvGPIO_Open(E_PT2, 0x01, E_IO_OUTPUT);    //set the operation mode of PT2.0
DrvGPIO_Open(E_PT2, 0x02, E_IO_INPUT);    //set the operation mode of PT2.1
```

5.3.2. DrvGPIO_Close

- **Prototype**

int32_t DrvGPIO_Close (E_DRVGPIO_PORT port, int32_t i32Bit, E_DRVGPIO_IO mode)

- **Description**

Close the specified operation mode of the specified GPIO pin

Configure the register

PT1　0x40800[23:16] / 0x40800[7:0] / 0x40804[23:16] / 0x40010[23:16]

© 2021-2022HYCON Technology Corp

PT2 0x40810[23:16] / 0x40810[7:0] / 0x40814[23:16] / 0x40014[23:16]

PT3 0x40820[23:16] / 0x40820[7:0] / 0x40824[23:16] / 0x40010[23:16]

- **Parameters**

port [in] : Specify GPIO port, , the effectively input range is 1~3

1：PT1 2：PT2 3：PT3.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be close if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

mode [in] : set the operation mode of the specified GPIO pin

0: input mode 1: output mode

2: pull up internally 3: external interrupt

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* close PT2.0 to GPIO output mode and PT2.1 to GPIO input mode*/ */
DrvGPIO_Close(E_PT2, 0x01, E_IO_OUTPUT); //close the specified operation mode of PT2.0
DrvGPIO_Close(E_PT2, 0x02, E_IO_INPUT); //close the specified operation mode of PT2.1
```

5.3.3. DrvGPIO_SetBit

- **Prototype**

unsigned int DrvGPIO_SetBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- **Description**

Set the output status value of the specified GPIO(PT1~PT3) pad to 1.

Configure the register 0x40804[7:0]/0x40814[7:0]/0x40824[7:0]

- **Parameters**

uport [in] : specify GPIO port, the effectively input range is 1~3.

1：PT1 2：PT2 3：PT3.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~7.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* configure PT2.0 as GPIO output mode*/
```

```
DrvGPIO_Open(E_PT2, 1, E_IO_OUTPUT);
/* Set PT2.0 to 1(high) */
DrvGPIO_SetBit(E_PT2,0);
```

5.3.4. DrvGPIO_ClrBit

- **Prototype**

  unsigned int DrvGPIO_ClrBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- **Description**

  Clear the output status value of the specified GPIO(PT1~PT3) port.

  Clear the register 0x40804[7:0] / 0x40814[7:0] / 0x40824[7:0]

- **Parameters**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　2：PT2　3：PT3.

  i32Bit [in] : Specify pin of the GPIO port. It could be 0~7.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/*Set PT1.0 output 0 */
DrvGPIO_ClrBit(E_PT1, 0);
```

5.3.5. DrvGPIO_GetBit

- **Prototype**

  unsigned int DrvGPIO_GetBit (E_DRVGPIO_PORT port, uint8_t u32Bit)

- **Description**

  Get the pin value from the specified input GPIO(PT1~PT3) port.

  Read the register 0x40808[7:0]/0x40818[7:0]/0x40828[7:0]

- **Parameters**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　2：PT2　3：PT3.

  i32Bit [in] : Specify pin of the GPIO port. The input range is 0~7.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0: Operation successful

0xff000000: Incorrect argument

- **Example**

```
uint32_t i32BitValue;
/* Configure PT2.1 as GPIO input mode, and read the input value of status*/
DrvGPIO_Open(E_PT2, 0x02, E_IO_INPUT);
DrvGPIO_Open(E_PT2, 0x02, E_IO_PullHigh);
i32Bit = DrvGPIO_GetBit(E_PT2,1);    // Read 0x40818[1]
```

5.3.6. DrvGPIO_SetPortBits

- **Prototype**

unsigned int DrvGPIO_SetPortBits (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **Description**

Set the output port value to the specified GPIO(PT1~PT3) port.

Configure the register 0x40804[7:0]/0x40814[7:0]/0x40824[7:0]

- **Parameters**

uport [in] : specify GPIO port, the effectively input range is 1~3.

1：PT1　2：PT2　3：PT3.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~7.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set PT2.1 and PT2.4 to 1*/
DrvGPIO_SetPortBits(E_PT2, 0x12);    //set 0x40814[1][4]
```

5.3.7. DrvGPIO_ClrPortBits

- **Prototype**

unsigned int DrvGPIO_ClrPortBits (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **Description**

Clear the output port value to the specified GPIO(PT1~PT3) port

Clear the register 0x40804[7:0] / 0x40814[7:0] / 0x40824[7:0]

- **Parameters**

uport [in] : specify GPIO port, the effectively input range is 1~3.

1：PT1　2：PT2　3：PT3.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~7.

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Clear the output value of PT2.1/PT2.4 to 0*/
DrvGPIO_ClrPortBits(E_PT2, 0x12);    //Clear 0x40814[1][4]
```

5.3.8. DrvGPIO_GetPortBits

- **Prototype**

  uint32_t DrvGPIO_GetPortBits (E_DRVGPIO_PORT port)

- **Description**

  Get the input port value from the specified GPIO(PT1~PT3) port.

  Read the register 0x40808[7 :0] / 0x40818[7 :0] / 0x40828[7 :0]

- **Parameters**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　2：PT2　3：PT3.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0: Operation successful

  0xff000000: Incorrect argument

- **Example**

```
uint32_t    i32Port;
i32Port = DrvGPIO_GetPortBits(E_PT2);    //Read 0x40818[7:0]
i32Port = DrvGPIO_GetPortBits(E_PT3);    // Read 0x40828[7:0]
```

5.3.9. DrvGPIO_IntTrigger

- **Prototype**

  int32_t DrvGPIO_IntTrigger (E_DRVGPIO_PORT port, uint32_t u32Bit, E_DRVGPIO_TriMethod mode)

- **Description**

  Set the specified interrupt pin to the specified interrupt trigger method operation mode.

  Configure the register 0x4080C[31:0]/0x4081C[31:0]

- **Parameters**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　2：PT2　3：PT3

u32Bit [in] : Specify pin of the GPIO port.

The bit will be set if the bit of the u32Bit is equal to 1. It could be 0~255.

mode [in] : set the specified interrupt method

0: disable the IO external interrupt trigger 1:rising-edge trigger

2: falling-edge trigger 3: level change trigger

4: level low trigger 5:level high trigger

6: level low trigger 7:level high trigger

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Configure PT2.0 to GPIO Interrupt mode ,trigger method is negative edge*/
DrvGPIO_Open(E_PT2, 0x01, E_IO_ IntEnable); //enable PT2 external interrupt. PT2.0.
DrvGPIO_IntTrigger(E_PT2, 0x01, E_N_Edge); //set PT2.0 interrupt trigger method. PT2.0.
```

5.3.10. DrvGPIO_ClearIntFlag

- **Prototype**

unsigned int DrvGPIO_ClearIntFlag (E_DRVGPIO_PORT port, uint32_t u32Bit)

- **Description**

Clear external interrupt flag.

Clear the register 0x40010[7:0] / 0x40014[7:0]

- **Parameters**

uport [in] : specify GPIO port, the effectively input range is 1~3.

1：PT1 2：PT2 3：PT3.

u32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF

The corresponding bit of register will be clear if the bit of the u32Bit is equal to 1.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

The current state of GPIO external interrupt 4flags

- **Example**

```
/* Clear PT2.2 interrupt flag */
DrvGPIO_ClearIntFlag(E_PT2, 0x04);    //0x40014[3]=0b
/* Clear PT2.3 interrupt flag*/
DrvGPIO_ClearIntFlag(E_PT2, 0x08);    //0x40014[7]=0b
```

© 2021-2022HYCON Technology Corp

5.3.11. DrvGPIO_GetIntFlag

- **Prototype**

  unsigned int DrvGPIO_GetIntFlag(E_DRVGPIO_PORT port)

- **Description**

  Get the port value from the specified Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

  Read the register 0x40010[7 :0] / 0x40014[7 :0].

- **Parameters**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

  1：PT1　2：PT2　3：PT3.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  The port value of the specified register: 0 ~ 0Xff

- **Example**

```
/* Get PT1 interrupt status. */
unsigned char  flag；flag=DrvGPIO_GetIntFlag(E_PT1);
```

5.3.12. DrvGPIO_PortIDIF

- **Prototype**

  unsigned int DrvGPIO_PortIDIF (uint32_t port)

- **Description**

  Read the PT1/PT2/PT3 condition flag of interrupt trigger when be in the external interrupt. The value depend on the interrupt trigger way. User must check the condition flag in register 0x4080C/0x4081C[31:24] to make sure the IO PIN on the corresponding initial status before get into the low power mode which waky up by the external interrupt.

  Operate the PT1 register 0x4080C[31:24] ,PT2 register 0x4081C[31:24] ,PT3 register 0x4082C[31:24]。

- **Parameters**

  uport [in] : specify GPIO port, the effectively input range is 1~3.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  The port value of the specified register: 0 ~ 0Xff

- **Example**

```
/* Get PT1 interrupt status. */
unsigned char  flag；flag=DrvGPIO_GetIntFlag(E_PT1);
```

© 2021-2022HYCON Technology Corp

5.3.13. DrvGPIO_LCDIOOpen

- **Prototype**

unsigned char DrvGPIO_LCDIOOpen (E_DRVGPIO_PORT port, int32_t i32Bit, E_DRVGPIO_IO mode)

- **Description**

Set the specified GPIO(PT6~PT13) pin to the specified GPIO operation mode(input or output).

Configure the register

PT6 0x40850[19:18][3:2]/ 0x40854[19:18][3:2]/ 0x40858[19:18][3:2] / 0x4085C[19:18][3:2]

PT7 0x40860[19:18][3:2]/ 0x40864[19:18][3:2]/ 0x40868[19:18][3:2] / 0x4086C[19:18][3:2]

PT8 0x40870[19:18][3:2]/ 0x40874[19:18][3:2]/ 0x40878[19:18][3:2] / 0x4087C[19:18][3:2]

PT9 0x40880[19:18][3:2]/ 0x40884[19:18][3:2]/ 0x40888[19:18][3:2] / 0x4088C[19:18][3:2]

PT10 0x40890[19:18][3:2]/ 0x40894[19:18][3:2]/ 0x40898[19:18][3:2] / 0x4089C[19:18][3:2]

PT13 0x408C0[19:18][3:2]/ 0x408C4[19:18][3:2]/ 0x408C8[19:18][3:2] / 0x408CC[19:18][3:2]

- **Parameters**

port [in] : specify GPIO port, the effectively input range is 0~7.

0：PT6　1：PT7　2：PT8　3：PT9

4：PT10　5：Rev　6：Rev　7：PT13.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be set if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

mode [in] : Set the operation mode of the specified GPIO pin

0: Enable input mode

1: Enable output mode

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* configure PT6.0 to GPIO output mode and PT6.1 to GPIO input mode*/
DrvGPIO_LCDIOOpen (E_PT6, 0x01, E_IO_OUTPUT); //set the operation mode of PT6.0, 0x40850[3]=1b
DrvGPIO_LCDIOOpen (E_PT6, 0x02, E_IO_INPUT); //set the operation mode of PT6.1, , 0x40850[18]=1b
```

5.3.14. DrvGPIO_LCDIOClose

- **Prototype**

unsigned char DrvGPIO_LCDIOClose (E_DRVGPIO_PORT port, int32_t i32Bit, E_DRVGPIO_IO mode)

- **Description**

Disable the specified GPIO operation mode(input or output) of the specified GPIO(PT6~PT13) pin.

© 2021-2022HYCON Technology Corp

Configure the register

PT6 0x40850[19:18][3:2]/ 0x40854[19:18][3:2]/ 0x40858[19:18][3:2] / 0x4085C[19:18][3:2]

PT7 0x40860[19:18][3:2]/ 0x40864[19:18][3:2]/ 0x40868[19:18][3:2] / 0x4086C[19:18][3:2]

PT8 0x40870[19:18][3:2]/ 0x40874[19:18][3:2]/ 0x40878[19:18][3:2] / 0x4087C[19:18][3:2]

PT9 0x40880[19:18][3:2]/ 0x40884[19:18][3:2]/ 0x40888[19:18][3:2] / 0x4088C[19:18][3:2]

PT10 0x40890[19:18][3:2]/ 0x40894[19:18][3:2]/ 0x40898[19:18][3:2] / 0x4089C[19:18][3:2]

PT13 0x408C0[19:18][3:2]/ 0x408C4[19:18][3:2]/ 0x408C8[19:18][3:2] / 0x408CC[19:18][3:2]

- **Parameters**

port [in] : specify GPIO port, the effectively input range is 0~7.

0：PT6　1：PT7　2：PT8　3：PT9

4：PT10　5：Rev　6：Rev　7：PT13.

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be set if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

mode [in] : Set the operation mode of the specified GPIO pin

0: Enable input mode

1: Enable output mode

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
DrvGPIO_LCDIOClose(E_PT6, 0x01, E_IO_OUTPUT); //close the operation mode of PT6.0
DrvGPIO_LCDIOClose(E_PT6, 0x02, E_IO_INPUT); //close the operation mode of PT6.1
```

5.3.15. DrvGPIO_LCDIOSetPorts

- **Prototype**

unsigned char DrvGPIO_LCDIOSetPorts (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **Description**

Set the output status value of the specified GPIO(PT6~PT13) to 1

Configure the register

PT6 0x40850[17][1]/ 0x40854[17][1]/ 0x40858[17][1] / 0x4085C[17][1]

PT7 0x40860[17][1]/ 0x40864[17][1]/ 0x40868[17][1] / 0x4086C[17][1]

PT8 0x40870[17][1]/ 0x40874[17][1]/ 0x40878[17][1] / 0x4087C[17][1]

PT9 0x40880[17][1]/ 0x40884[17][1]/ 0x40888[17][1] / 0x4088C[17][1]

PT10 0x40890[17][1]/ 0x40894[17][1]/ 0x40898[17][1] / 0x4089C[17][1]

PT13 0x408C0[17][1]/ 0x408C4[17][1]/ 0x408C8[17][1] / 0x408CC[17][1]

© 2021-2022HYCON Technology Corp

- **Parameters**

  port [in] : Specify GPIO port, the effectively input range is 0~7.

  0 : PT6 1 : PT7 2 : PT8 3 : PT9

  4 : PT10 5 : Rev 6 : Rev 7 : PT13.

  ui32Data [in] : The specified pin of the GPIO port. It could be 0~0xFF.

  The operation mode of the pin will be set if the bit of i32Bit is equal to 1

  The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* configure PT6.1 and PT6.4 to 1 */
DrvGPIO_LCDIOSetPorts(E_PT6, 0x12);
```

5.3.16. DrvGPIO_LCDIOClrPorts

- **Prototype**

  unsigned char DrvGPIO_LCDIOClrPorts (E_DRVGPIO_PORT uport, unsigned int ui32Data)

- **Description**

  Set the output status value of the specified GPIO(PT6~PT13) to 0

  Configure the register

  PT6 0x40850[17][1]/ 0x40854[17][1]/ 0x40858[17][1] / 0x4085C[17][1]

  PT7 0x40860[17][1]/ 0x40864[17][1]/ 0x40868[17][1] / 0x4086C[17][1]

  PT8 0x40870[17][1]/ 0x40874[17][1]/ 0x40878[17][1] / 0x4087C[17][1]

  PT9 0x40880[17][1]/ 0x40884[17][1]/ 0x40888[17][1] / 0x4088C[17][1]

  PT10 0x40890[17][1]/ 0x40894[17][1]/ 0x40898[17][1] / 0x4089C[17][1]

  PT13 0x408C0[17][1]/ 0x408C4[17][1]/ 0x408C8[17][1] / 0x408CC[17][1]

- **Parameters**

  port [in] : Specify GPIO port, , the effectively input range is 0~7.

  0 : PT6 1 : PT7 2 : PT8 3 : PT9

  4 : PT10 5 : Rev 6 : Rev 7 : PT13.

  ui32Data [in] : The specified pin of the GPIO port. It could be 0~0xFF.

  The operation mode of the pin will be set if the bit of i32Bit is equal to 1

  The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

© 2021-2022HYCON Technology Corp

0: Operation successful

Other : Incorrect argument

- **Example**

/* configure PT6.1 and PT6.4 */

DrvGPIO_LCDIOClrPorts(E_PT6, 0x12);

5.3.17. DrvGPIO_LCDIOSetBit

- **Prototype**

unsigned char DrvGPIO_LCDIOSetBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

- **Description**

Set the output status value of the specified GPIO(PT6~PT13) to 1

Configure the register

PT6 0x40850[17][1]/ 0x40854[17][1]/ 0x40858[17][1] / 0x4085C[17][1]

PT7 0x40860[17][1]/ 0x40864[17][1]/ 0x40868[17][1] / 0x4086C[17][1]

PT8 0x40870[17][1]/ 0x40874[17][1]/ 0x40878[17][1] / 0x4087C[17][1]

PT9 0x40880[17][1]/ 0x40884[17][1]/ 0x40888[17][1] / 0x4088C[17][1]

PT10 0x40890[17][1]/ 0x40894[17][1]/ 0x40898[17][1] / 0x4089C[17][1]

PT13 0x408C0[17][1]/ 0x408C4[17][1]/ 0x408C8[17][1] / 0x408CC[17][1]

- **Parameters**

port [in] : Specify GPIO port, , the effectively input range is 0~7.

0：PT6　1：PT7　2：PT8　3：PT9

4：PT10　5：Rev　6：Rev　7：PT13.

i32Bit [in] : The specified pin of the GPIO port. It could be 0~7.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/* configure PT6.0 as output mode*/

DrvGPIO_LCDIOOpen (E_PT6, 1, E_IO_OUTPUT); //0x40850[3]=1b

/* configure PT6.0 output 1 */

DrvGPIO_LCDIOSetBit(E_PT6, 0);　0x40850[1]=1b

5.3.18. DrvGPIO_LCDIOClrBit

- **Prototype**

unsigned char DrvGPIO_LCDIOClrBit (E_DRVGPIO_PORT uport, unsigned int i32Bit)

© 2021-2022HYCON Technology Corp

- **Description**

Set the output status value of the specified GPIO(PT6~PT13) to 0

Configure the register

PT6 0x40850[17][1]/ 0x40854[17][1]/ 0x40858[17][1] / 0x4085C[17][1]

PT7 0x40860[17][1]/ 0x40864[17][1]/ 0x40868[17][1] / 0x4086C[17][1]

PT8 0x40870[17][1]/ 0x40874[17][1]/ 0x40878[17][1] / 0x4087C[17][1]

PT9 0x40880[17][1]/ 0x40884[17][1]/ 0x40888[17][1] / 0x4088C[17][1]

PT10 0x40890[17][1]/ 0x40894[17][1]/ 0x40898[17][1] / 0x4089C[17][1]

PT13 0x408C0[17][1]/ 0x408C4[17][1]/ 0x408C8[17][1] / 0x408CC[17][1]

- **Parameters**

port [in] : Specify GPIO port, , the effectively input range is 0~7.

0：PT6　1：PT7　2：PT8　3：PT9

4：PT10　5：Rev　6：Rev　7：PT13.

i32Bit [in] : The specified pin of the GPIO port. It could be 0~7.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* configure PT6.0 output 0 */
DrvGPIO_LCDIOClrBit(E_PT6, 0);
```

5.3.19. DrvGPIO_LCDIOGetPorts

- **Prototype**

unsigned char DrvGPIO_LCDIOGetPorts (E_DRVGPIO_PORT port)

- **Description**

Get the input port value from the specified GPIO(PT6~PT13) port.

Read the register

PT6 0x40850[16][0]/ 0x40854[16][0]/ 0x40858[16][0] / 0x4085C[16][0]

PT7 0x40860[16][0]/ 0x40864[16][0]/ 0x40868[16][0] / 0x4086C[16][0]

PT8 0x40870[16][0]/ 0x40874[16][0]/ 0x40878[16][0] / 0x4087C[16][0]

PT9 0x40880[16][0]/ 0x40884[16][0]/ 0x40888[16][0] / 0x4088C[16][0]

PT10 0x40890[16][0]/ 0x40894[16][0]/ 0x40898[16][0] / 0x4089C[16][0]

PT13 0x408C0[16][0]/ 0x408C4[16][0]/ 0x408C8[16][0] / 0x408CC[16][0]

- **Parameters**

port [in] : Specify GPIO port, , the effectively input range is 0~7.

0：PT6　1：PT7　2：PT8　3：PT9

4：PT10　5：Rev　6：Rev　7：PT13.

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0 ~ 0xFF :The specified input port valu

  0xff000000: Incorrect argument.

- **Example**

```
/* Get the PT6 port input data value */
uint32_t    i32Port; i32Port = DrvGPIO_LCDIOGetPorts(E_PT6);
```

5.3.20. DrvGPIO_LCDIOGetBit

- **Prototype**

  unsigned int DrvGPIO_LCDIOGetBit (E_DRVGPIO_PORT port, uint8_t u32Bit)

- **Description**

  Get the input port value from the specified GPIO(PT6~PT10) port.

  Read the register

  PT6 0x40850[16][0]/ 0x40854[16][0]/ 0x40858[16][0] / 0x4085C[16][0]

  PT7 0x40860[16][0]/ 0x40864[16][0]/ 0x40868[16][0] / 0x4086C[16][0]

  PT8 0x40870[16][0]/ 0x40874[16][0]/ 0x40878[16][0] / 0x4087C[16][0]

  PT9 0x40880[16][0]/ 0x40884[16][0]/ 0x40888[16][0] / 0x4088C[16][0]

  PT10 0x40890[16][0]/ 0x40894[16][0]/ 0x40898[16][0] / 0x4089C[16][0]

  PT13 0x408C0[16][0]/ 0x408C4[16][0]/ 0x408C8[16][0] / 0x408CC[16][0]

- **Parameters**

  port [in] : Specify GPIO port, , the effectively input range is 0~7.

  0：PT6　1：PT7　2：PT8　3：PT9

  4：PT10　5：Rev　6：Rev　7：PT13.

  i32Bit [in] : The specified pin of the GPIO port. It could be 0~7.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0/1 :The specified input port value

  0xff: Incorrect argument

- **Example**

```
uint32_t i32BitValue;
/* Read PT6.0~PT6.7 input status */
uint32_t    i32Bit;
i32Bit = DrvGPIO_LCDIOGetBit(E_PT6,0);    // read 0x40850[0]
i32Bit = DrvGPIO_LCDIOGetBit(E_PT6,1);    // read 0x40850[16]
   i32Bit= DrvGPIO_LCDIOGetBit(E_PT6,2);    // read 0x40854[0]
```

© 2021-2022HYCON Technology Corp

```
i32Bit= DrvGPIO_LCDIOGetBit(E_PT6,3);   // read 0x40854[16]
i32Bit= DrvGPIO_LCDIOGetBit(E_PT6,4);   // read 0x40858[0]
i32Bit= DrvGPIO_LCDIOGetBit(E_PT6,5);   // read 0x40858[16]
i32Bit= DrvGPIO_LCDIOGetBit(E_PT6,6);   // read 0x4085C[0]
i32Bit= DrvGPIO_LCDIOGetBit(E_PT6,7);   // read 0x4085C[16]
```

5.3.21. DrvGPIO_EnableAnalogPin

- **Prototype**

unsigned char DrvGPIO_EnableAnalogPin(short port,unsigned int i32Bit)

- **Description**

Close the digital operation mode of the specified GPIO pin, it could be input/output/external interrupt/pull-up/interrupt trigger edge and open analog operation mode

Configure the register

PT1 0x40800[23:16] / 0x40800[7:0] / 0x40804[23:16] /0x4080C[23:0]/ 0x40010[23:16]

PT2 0x40810[23:16] / 0x40810[7:0] / 0x40814[23:16] /0x4081C[23:0]/ 0x40014[23:16]

PT3 0x40820[23:16] / 0x40820[7:0] / 0x40824[23:16] /0x4082C[23:0]/ 0x40024[23:16]

- **Parameters**

port [in] : specify GPIO port, the effectively input range is 1~3

1：PT1　2：PT2　3：PT3

i32Bit [in] : Specify pin of the GPIO port. It could be 0~0xFF.

The operation mode of the pin will be change if the bit of i32Bit is equal to 1

The operation mode of the pin will not be change if the bit of i32Bit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0: Operation successful

1: Incorrect argument

- **Example**

```
/* Close PT3.1/PT3.3/PT3.5/PT3.7 digital operation mode*/
DrvGPIO_Open(E_PT3,0xAA,E_IO_INPUT);
DrvGPIO_Open(E_PT3,0x55,E_IO_OUTPUT);
DrvGPIO_Open(E_PT3,0xAA,E_IO_PullHigh);
DrvGPIO_IntTrigger(E_PT3,0xAA,E_N_Edge);
DrvGPIO_EnableAnalogPin(E_PT3,0xAA);
```

5.3.22. DrvGPIO_PT1_EnableINPUT

- **Prototype**

void DrvGPIO_PT1_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40804[23:16]

- **Parameters**

ubit[in] : specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT1.0/PT1.1 as input mode*/
DrvGPIO_PT1_EnableINPUT(0x01|0x02);
```

5.3.23. DrvGPIO_PT1_DisableINPUT

- **Prototype**

void DrvGPIO_PT1_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40804[23:16]

- **Parameters**

ubit[in] : specified PT1 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the input mode of PT1.0/PT1.1*/
DrvGPIO_PT1_DisableINPUT(0x01|0x02);
```

5.3.24. DrvGPIO_PT1_EnablePullHigh

- **Prototype**

void DrvGPIO_PT1_EnablePullHigh(short int ubit)

- **Description**

  Enable the pull up of the specified GPIO pin .

  Configure the register 0x40800[23:16]

- **Parameters**

  ubit[in] : specified PT1 pin. It could be 0~0xff

  Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* enable the pull up of PT1.0/PT1.1 */

  DrvGPIO_PT1_EnablePullHigh(0x01|0x02);

5.3.25. DrvGPIO_PT1_DisablePullHigh

- **Prototype**

  void DrvGPIO_PT1_DisablePullHigh(short int ubit)

- **Description**

  Disable the pull up of the specified GPIO pin .

  Configure the register 0x40800[23:16]

- **Parameters**

  ubit[in] specified PT1 pin. It could be 0~0xff

  Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* disable the pull up of PT1.0/PT1.1 */

  DrvGPIO_PT1_DisablePullHigh(0x01|0x02);

5.3.26. DrvGPIO_PT1_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT1_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

© 2021-2022HYCON Technology Corp

Configure the register 0x40800[7:0]

- **Parameters**

ubit[in] specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT1.0/PT1.1 as output mode*/
DrvGPIO_PT1_EnableOUTPUT(0x01|0x02);
```

5.3.27. DrvGPIO_PT1_DisableOUTPUT

- **Prototype**

void DrvGPIO_PT1_DisableOUTPUT(short int ubit)

- **Description**

Disable the output mode of the specified GPIO pin .

Configure the register 0x40800[7:0]

- **Parameters**

ubit[in] specified PT1 pin. It could be 0~0xff

Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the output mode of PT1.0/PT1.1*/
DrvGPIO_PT1_DisableOUTPUT(0x01|0x02);
```

5.3.28. DrvGPIO_PT1_EnableINT

- **Prototype**

void DrvGPIO_PT1_EnableINT(short int ubit)

- **Description**

Enable the external interrupt of the specified GPIO pin .

Configure the register 0x40010[23:16]

© 2021-2022HYCON Technology Corp

- **Parameters**

  ubit[in] : specified PT1 pin. It could be 0~0xff

  Set the specified GPIO pin to enable the external interrupt if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* enable the external interrupt of PT1.0/PT1.1 */
DrvGPIO_PT1_EnableINT(0x01|0x02);
```

5.3.29. DrvGPIO_PT1_DisableINT

- **Prototype**

  void DrvGPIO_PT1_DisableINT(short int ubit)

- **Description**

  Disable the external interrupt of the specified GPIO pin .

  Configure the register 0x40010[23:16]

- **Parameters**

  ubit[in] : specified PT1 pin. It could be 0~0xff

  Set the specified GPIO pin to disable the external interrupt if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* disable the external interrupt of PT1.0/PT1.1 */
DrvGPIO_PT1_DisableINT(0x01|0x02);
```

5.3.30. DrvGPIO_PT1_IntTriggerPorts

- **Prototype**

  void DrvGPIO_PT1_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

- **Description**

  Enable the external interrupt trigger of the specified GPIO pin . Select the method of interrupt trigger.

  Configure the register 0x4080c[31:0]

- **Parameters**

  u32Bit [in] : specified PT1 pin. It could be 0~0xff

© 2021-2022HYCON Technology Corp

Set the specified GPIO if the bit of u32Bit is equal to 1

Disable the interrupt trigger of specified GPIO pin if the bit of u32Bit is equal to 0

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger	1：positive edge	2：negative edge	3：level change
4：low level	5：high level	6：low level	7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set the negative edge as the interrupt method of PT1.0*/
DrvGPIO_PT1_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT1_IntTriggerPorts(0x1, E_N_Edge);    //configure the interrupt trigger method
```

5.3.31. DrvGPIO_PT1_IntTriggerBit

- **Prototype**

void DrvGPIO_PT1_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

- **Description**

Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

Configure the register 0x4080c[31:0]

- **Parameters**

u32Bit [in] : specified PT1 pin. It could be 0~7 stand for bit7~bit0 of GPIO port

The specified GPIO pin will be set.

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger	1：positive edge	2：negative edge	3：level change
4：low level	5：high level	6：low level	7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set the negative edge as the interrupt method of PT1.0*/
DrvGPIO_PT1_EnableINT(0x1);// enable GPIO external interrupt
DrvGPIO_PT1_IntTriggerPorts(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.32. DrvGPIO_PT1_GetIntFlag

- **Prototype**

unsigned char DrvGPIO_PT1_GetIntFlag(void)

© 2021-2022HYCON Technology Corp

- **Description**

Get the port value from the PT1 Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

Read the register 0x40010[7:0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

The interrupt flag value of the specified register: 0 ~ 0xff

- **Example**

```
/* Get PT1 interrupt flag. */
unsigned char    flag ; flag=DrvGPIO_PT1_GetIntFlag();
```

5.3.33. DrvGPIO_PT1_ClearIntFlag

- **Prototype**

void DrvGPIO_PT1_ClearIntFlag(short int uint32)

- **Description**

Clear the external interrupt flag of PT1

Configure the register 0X40010[7:0]

- **Parameters**

u32Bit [in] : specified PT1 pin. It could be 0~0xff

The each bit of u32Bit is corresponding to one pin .

Clear the corresponding interrupt flag when the specified bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* Clear PT1.2 interrupt flag */
DrvGPIO_PT1_ClearIntFlag(0x04);
/* Clear PT1.3 interrupt flag*/
DrvGPIO_PT1_ClearIntFlag(0x08);
```

5.3.34. DrvGPIO_PT1_GetPortBits

- **Prototype**

unsigned char DrvGPIO_PT1_GetPortBits (void)

© 2021-2022HYCON Technology Corp

- **Description**

  Get the input port data from the specified GPIO port.

  Read the register 0x40808[7:0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0 ~ 0xFF :The input value of specified port

- **Example**

  /* Get the PT1 port input data value */

  uint32_t i32Port; i32Port = DrvGPIO_PT1_GetPortBits();

5.3.35. DrvGPIO_PT1_SetPortBits

- **Prototype**

  void DrvGPIO_PT1_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Configure the register 0x40804[7:0]

- **Parameters**

  i32Data [in] : specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* Set PT1.2, PT1.4 */

  DrvGPIO_PT1_SetPortBits(0x14);

5.3.36. DrvGPIO_PT1_ClrPortBits

- **Prototype**

  void DrvGPIO_PT1_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Configure the register 0x40804[7:0]

- **Parameters**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* clear PT1.1    PT1.4 */
DrvGPIO_PT1_ClrPortBits(0x12);
```

5.3.37. DrvGPIO_PT2_EnableINPUT

- **Prototype**

  void DrvGPIO_PT2_EnableINPUT(short int ubit)

- **Description**

  Enable the input mode of the specified GPIO pin .

  Configure the register 0x40814[23:16]

- **Parameters**

  ubit[in] : specified PT2 pin. It could be 0~0xff

  Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* set PT2.0/PT2.1 as input mode*/
DrvGPIO_PT2_EnableINPUT(0x01|0x02);
```

5.3.38. DrvGPIO_PT2_DisableINPUT

- **Prototype**

  void DrvGPIO_PT2_DisableINPUT(short int ubit)

- **Description**

  Disable the input mode of the specified GPIO pin .

  Configure the register 0x40814[23:16]

- **Parameters**

ubit[in] : specified PT2 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the input mode of PT2.0/PT2.1*/
DrvGPIO_PT2_DisableINPUT(0x01|0x02);
```

5.3.39. DrvGPIO_PT2_EnablePullHigh

- **Prototype**

void DrvGPIO_PT2_EnablePullHigh(short int ubit)

- **Description**

Enable the pull up of the specified GPIO pin .

Configure the register 0x40810[23:16]

- **Parameters**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to enable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* enable the pull up of PT2.0/PT2.1 */
DrvGPIO_PT2_EnablePullHigh(0x01|0x02);
```

5.3.40. DrvGPIO_PT2_DisablePullHigh

- **Prototype**

void DrvGPIO_PT2_DisablePullHigh(short int ubit)

- **Description**

Disable the pull up of the specified GPIO pin .

Configure the register 0x40810[23:16]

- **Parameters**

ubit[in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

© 2021-2022HYCON Technology Corp

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* disable the pull up of PT2.0/PT2.1 */
DrvGPIO_PT2_DisablePullHigh(0x01|0x02);
```

5.3.41. DrvGPIO_PT2_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT2_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

  Configure the register 0x40810[7:0]

- **Parameters**

  ubit[in] : specified PT2 pin. It could be 0~0xff

  Set the specified GPIO pin to the output operation mode if the bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* set PT2.0/PT2.1 as output mode*/
DrvGPIO_PT2_EnableOUTPUT(0x01|0x02);
```

5.3.42. DrvGPIO_PT2_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT2_DisableOUTPUT(short int ubit)

- **Description**

  Disable the output mode of the specified GPIO pin .

  Configure the register 0x40810[7:0]

- **Parameters**

  ubit[in] : specified PT2 pin. It could be 0~0xff

  Disable the output mode of the specified GPIO pin if the bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* disable the output mode of PT2.0/PT2.1*/

  DrvGPIO_PT2_DisableOUTPUT(0x01|0x02);

5.3.43. DrvGPIO_PT2_EnableINT

- **Prototype**

  void DrvGPIO_PT2_EnableINT(short int ubit)

- **Description**

  Enable the external interrupt of the specified GPIO pin .

  Configure the register 0x40014[23:16]

- **Parameters**

  ubit[in] : specified PT2 pin. It could be 0~0xff

  Set the specified GPIO pin to enable the external interrupt if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* enable the external interrupt of PT2.0/PT2.1 */

  DrvGPIO_PT2_EnableINT(0x01|0x02);

5.3.44. DrvGPIO_PT2_DisableINT

- **Prototype**

  void DrvGPIO_PT2_DisableINT(short int ubit)

- **Description**

  Disable the external interrupt of the specified GPIO pin .

  Configure the register 0x40014[23:16]

- **Parameters**

  ubit[in] : specified PT2 pin. It could be 0~0xff

  Set the specified GPIO pin to disable the external interrupt if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

© 2021-2022HYCON Technology Corp

● **Include**

Peripheral_lib/DrvGPIO.h

● **Return Vaule**

None

● **Example**

```
/* disable the external interrupt of PT2.0/PT2.1 */
DrvGPIO_PT2_DisableINT(0x01|0x02);
```

5.3.45. DrvGPIO_PT2_IntTriggerPorts

● **Prototype**

void DrvGPIO_PT2_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

● **Description**

Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

Configure the register 0x4081C[31:0]

● **Parameters**

u32Bit [in] : specified PT2 pin. It could be 0~0xff

Set the specified GPIO if the bit of u32Bit is equal to 1

Disable the interrupt trigger of specified GPIO pin if the bit of u32Bit is equal to 0

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger 1：rising-edge 2：falling-edge 3：level change

4：low level 5：high level 6：low level 7：high level

● **Include**

Peripheral_lib/DrvGPIO.h

● **Return Vaule**

None

● **Example**

```
/* set the falling edge as the interrupt method of PT2.0*/
DrvGPIO_PT2_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT2_IntTriggerPorts(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.46. DrvGPIO_PT2_IntTriggerBit

● **Prototype**

void DrvGPIO_PT2_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

● **Description**

Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

Configure the register 0x4081C[31:0]

● **Parameters**

© 2021-2022HYCON Technology Corp

u32Bit [in] : specified PT2 pin. It could be 0~7 stand for bit7~bit0 of GPIO port

The specified GPIO pin will be set.

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger 1：rising-edge 2：falling-edge 3：level change

4：low level 5：high level 6：low level 7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* set the falling-edge as the interrupt method of PT2.0*/

DrvGPIO_PT2_EnableINT(0x1); // enable GPIO external interrupt

DrvGPIO_PT2_IntTriggerBit(0x1, E_N_Edge); //configure the interrupt trigger method

5.3.47. DrvGPIO_PT2_GetIntFlag

- **Prototype**

unsigned char DrvGPIO_PT2_GetIntFlag(void)

- **Description**

Get the port value from the PT2 Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

Read the register 0x40014[7:0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

The interrupt flag value of the specified register: 0 ~ 0xff

- **Example**

/* Get PT2 interrupt flag. */

unsigned char flag；flag=DrvGPIO_PT2_GetIntFlag();

5.3.48. DrvGPIO_PT2_ClearIntFlag

- **Prototype**

void DrvGPIO_PT2_ClearIntFlag(short int uint32)

- **Description**

Clear the external interrupt flag of PT2

© 2021-2022HYCON Technology Corp

Configure the register 0x40014[7 :0]

- **Parameters**

u32Bit [in] : specified PT2 pin. It could be 0~0xff

The each bit of u32Bit corresponding to one pin .

Clear the corresponding interrupt flag when the specified bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* Clear PT2.2 interrupt flag */
DrvGPIO_PT2_ClearIntFlag(0x04);
/* Clear PT2.3 interrupt flag*/
DrvGPIO_PT2_ClearIntFlag(0x08);
```

5.3.49. DrvGPIO_PT2_GetPortBits

- **Prototype**

unsigned char DrvGPIO_PT2_GetPortBits (void)

- **Description**

Get the input port value from the specified GPIO port.

Read the register 0x40818[7:0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0 ~ 0xFF :The input value of specified port

- **Example**

```
/* Get the PT2 port input data value */
uint32_t i32Port; i32Port = DrvGPIO_PT2_GetPortBits();
```

5.3.50. DrvGPIO_PT2_SetPortBits

- **Prototype**

void DrvGPIO_PT2_SetPortBits (unsigned char ui32Data)

- **Description**

Set the output port value to the specified pin.

Read the register 0x40814[7:0]

- **Parameters**

i32Data [in] : specify which bit to be set. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* Set PT2.2, PT2.4 to 1(high) */
DrvGPIO_PT2_SetPortBits(0x14);
```

5.3.51. DrvGPIO_PT2_ClrPortBits

- **Prototype**

void DrvGPIO_PT2_ClrPortBits (unsigned int ui32Data)

- **Description**

Clear the output data of the specified pin.

Read the register 0x40814[7:0]

- **Parameters**

i32Data [in] : specify which bit to be clear. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* clear PT2.1, PT2.4 */
DrvGPIO_PT2_ClrPortBits(0x12);
```

5.3.52. DrvGPIO_PT3_EnableINPUT

- **Prototype**

void DrvGPIO_PT3_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40824[23:16]

- **Parameters**

© 2021-2022HYCON Technology Corp

ubit[in] : specified PT3 pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT3.0/PT3.1 as input mode*/
DrvGPIO_PT3_EnableINPUT(0x01|0x02);
```

5.3.53. DrvGPIO_PT3_DisableINPUT

- **Prototype**

void DrvGPIO_PT3_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40824[23:16]

- **Parameters**

ubit[in] : specified PT3 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the input mode of PT3.0/PT3.1*/
DrvGPIO_PT3_DisableINPUT(0x01|0x02);
```

5.3.54. DrvGPIO_PT3_EnablePullHigh

- **Prototype**

void DrvGPIO_PT3_EnablePullHigh(short int ubit)

- **Description**

Enable the pull up of the specified GPIO pin .

Configure the register 0x40820[23:16]

- **Parameters**

ubit[in] : specified PT3 pin. It could be 0~0xff

© 2021-2022HYCON Technology Corp

Set the specified GPIO pin to enable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* enable the pull up of PT3.0/PT3.1 */
DrvGPIO_PT3_EnablePullHigh(0x01|0x02);
```

5.3.55. DrvGPIO_PT3_DisablePullHigh

- **Prototype**

void DrvGPIO_PT3_DisablePullHigh(short int ubit)

- **Description**

Disable the pull up of the specified GPIO pin .

Configure the register 0x40820[23:16]

- **Parameters**

ubit[in] : specified PT3 pin. It could be 0~0xff

Set the specified GPIO pin to disable pull-up if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the pull up of PT3.0/PT3.1 */
DrvGPIO_PT3_DisablePullHigh(0x01|0x02);
```

5.3.56. DrvGPIO_PT3_EnableOUTPUT

- **Prototype**

void DrvGPIO_PT3_EnableOUTPUT(short int ubit)

- **Description**

Enable the output mode of the specified GPIO pin .

Configure the register 0x40820[7:0]

- **Parameters**

ubit[in] : specified PT3 pin. It could be 0~0xff

Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

© 2021-2022HYCON Technology Corp

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* set PT3.0/PT3.1 as output mode*/

DrvGPIO_PT3_EnableOUTPUT(0x01|0x02);

5.3.57. DrvGPIO_PT3_DisableOUTPUT

- **Prototype**

void DrvGPIO_PT3_DisableOUTPUT(short int ubit)

- **Description**

Disable the output mode of the specified GPIO pin .

Configure the register 0x40820[7:0]

- **Parameters**

ubit[in] : specified PT3 pin. It could be 0~0xff

Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* disable the output mode of PT3.0/PT3.1*/

DrvGPIO_PT3_DisableOUTPUT(0x01|0x02);

5.3.58. DrvGPIO_PT3_EnableINT

- **Prototype**

void DrvGPIO_PT3_EnableINT(short int ubit)

- **Description**

Enable the external interrupt of the specified GPIO pin .

Configure the register 0x40024[23:16]

- **Parameters**

ubit[in] : specified PT3 pin. It could be 0~0xff

Set the specified GPIO pin to enable the external interrupt if the bit of ubit is equal to 1

The status of specified GPIO pin would not change if the bit of ubit is equal to 0

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* enable the external interrupt of PT3.0/PT3.1 */

  DrvGPIO_PT3_EnableINT(0x01|0x02);

5.3.59. DrvGPIO_PT3_DisableINT

- **Prototype**

  void DrvGPIO_PT3_DisableINT(short int ubit)

- **Description**

  Disable the external interrupt of the specified GPIO pin .

  Configure the register 0x40024[23:16]

- **Parameters**

  ubit[in] : specified PT3 pin. It could be 0~0xff

  Set the specified GPIO pin to disable the external interrupt if the bit of ubit is equal to 1

  The status of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /*disable the external interrupt of PT3.0/PT3.1 */

  DrvGPIO_PT3_DisableINT(0x01|0x02);

5.3.60. DrvGPIO_PT3_IntTriggerPorts

- **Prototype**

  void DrvGPIO_PT3_IntTriggerPorts(uint32_t i32Bit, uint32_t mode)

- **Description**

  Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

  Configure the register 0x4082C[31:0]

- **Parameters**

  u32Bit [in] : specified PT3 pin. It could be 0~0xff

  Set the specified GPIO if the bit of u32Bit is equal to 1

  Disable the interrupt trigger of specified GPIO pin if the bit of u32Bit is equal to 0

  mode [in] : interrupt trigger method . it could be 0~7

  0：disable GPIO interrupt trigger　1：rising-edge　2：falling-edge 3：level change

© 2021-2022HYCON Technology Corp

4：low level 5：high level 6：low level 7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set the falling edge as the interrupt method of PT3.0 */
DrvGPIO_PT3_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT3_IntTriggerPorts(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.61. DrvGPIO_PT3_IntTriggerBit

- **Prototype**

void DrvGPIO_PT3_IntTriggerBit(uint32_t i32Bit, uint32_t mode)

- **Description**

Enable the external interrupt trigger of the specified GPIO pin . select the method of interrupt trigger.

Configure the register 0x4082C[31:0]

- **Parameters**

u32Bit [in] : specified PT3 pin. It could be 0~7 stand for bit7~bit0 of GPIO port

The specified GPIO pin will be set.

mode [in] : interrupt trigger method . it could be 0~7

0：disable GPIO interrupt trigger 1：rising-edge 2：falling-edge 3：level change

4：low level 5：high level 6：low level 7：high level

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set the falling-edge as the interrupt method of PT3.0*/
DrvGPIO_PT3_EnableINT(0x1); // enable GPIO external interrupt
DrvGPIO_PT3_IntTriggerBit(0x1, E_N_Edge); //configure the interrupt trigger method
```

5.3.62. DrvGPIO_PT3_GetIntFlag

- **Prototype**

unsigned char DrvGPIO_PT3_GetIntFlag(void)

- **Description**

Get the port value from the PT3 Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it is meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt

© 2021-2022HYCON Technology Corp

occurred at that bit.

Read the register0x40024[7:0].

- **Parameters**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

The interrupt flag value of the specified register: 0 ~ 0xff

- **Example**

```
/* Get PT3 interrupt flag. */
unsigned char  flag ; flag=DrvGPIO_PT3_GetIntFlag();
```

5.3.63. DrvGPIO_PT3_ClearIntFlag

- **Prototype**

void DrvGPIO_PT3_ClearIntFlag(short int uint32)

- **Description**

Clear the external interrupt flag of PT2

Configure the register 0x40024[7 :0] 。

- **Parameters**

u32Bit [in] : specified PT3 pin. It could be 0~0xff

The each bit of u32Bit corresponding to one pin .

Clear the corresponding interrupt flag when the specified bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* Clear PT3.2 interrupt flag */
DrvGPIO_PT3_ClearIntFlag(0x04);
/* Clear PT3.3 interrupt flag*/
DrvGPIO_PT3_ClearIntFlag(0x08);
```

5.3.64. DrvGPIO_PT3_GetPortBits

- **Prototype**

unsigned char DrvGPIO_PT3_GetPortBits (void)

- **Description**

Get the input port value from the specified GPIO port.

Read the register 0x40828[7 :0]

© 2021-2022HYCON Technology Corp

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0 ~ 0xFF :The input value of specified port

- **Example**

```
/* Get the PT3 port input data value */
uint32_t    i32Port; i32Port = DrvGPIO_PT3_GetPortBits();
```

5.3.65. DrvGPIO_PT3_SetPortBits

- **Prototype**

  void DrvGPIO_PT3_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Read the register 0x40824[7:0]

- **Parameters**

  i32Data [in] : specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/*Set PT3.2, PT3.4 to 1(high) */
DrvGPIO_PT3_SetPortBits(0x14);
```

5.3.66. DrvGPIO_PT3_ClrPortBits

- **Prototype**

  void DrvGPIO_PT3_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Read the register 0x40824[7:0]

- **Parameters**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

© 2021-2022HYCON Technology Corp

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* clear PT3.1/PT3.4 */
DrvGPIO_PT3_ClrPortBits(0x12);
```

5.3.67. DrvGPIO_PT6_EnableINPUT

- **Prototype**

void DrvGPIO_PT6_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40850[18][2]/ 0x40854[18][2]/ 0x40858[18][2] / 0x4085C[18][2]

- **Parameters**

ubit[in] : specified PT6pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT6.0/PT6.1 as input mode*/
DrvGPIO_PT6_EnableINPUT(0x01|0x02);
```

5.3.68. DrvGPIO_PT6_DisableINPUT

- **Prototype**

void DrvGPIO_PT6_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40850[18][2]/ 0x40854[18][2]/ 0x40858[18][2] / 0x4085C[18][2]

- **Parameters**

ubit[in] : specified PT6 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

© 2021-2022HYCON Technology Corp

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the input mode of PT6.0/PT6.1*/
DrvGPIO_PT6_DisableINPUT(0x01|0x02);
```

5.3.69. DrvGPIO_PT6_EnableOUTPUT

- **Prototype**

void DrvGPIO_PT6_EnableOUTPUT(short int ubit)

- **Description**

Enable the output mode of the specified GPIO pin .

Configure the register 0x40850[19][3]/ 0x40854[19][3]/ 0x40858[19][3] / 0x4085C[19][3]

- **Parameters**

ubit[in] : specified PT6 pin. It could be 0~0xff

Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT6.0/PT6.1 as output mode*/
DrvGPIO_PT6_EnableOUTPUT(0x01|0x02);
```

5.3.70. DrvGPIO_PT6_DisableOUTPUT

- **Prototype**

void DrvGPIO_PT6_DisableOUTPUT(short int ubit)

- **Description**

Disable the output mode of the specified GPIO pin .

Configure the register 0x40850[19][3]/ 0x40854[19][3]/ 0x40858[19][3] / 0x4085C[19][3]

- **Parameters**

ubit[in] : specified PT6 pin. It could be 0~0xff

Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* disable the output mode of PT6.0/PT6.1*/

  DrvGPIO_PT6_DisableOUTPUT(0x01|0x02);

5.3.71. DrvGPIO_PT6_GetPortBits

- **Prototype**

  unsigned char DrvGPIO_PT6_GetPortBits (void)

- **Description**

  Get the input port data from the specified GPIO port.

  Read the register 0x40850[16][0]/ 0x40854[16][0]/ 0x40858[16][0] / 0x4085C[16][0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0 ~ 0xFF :The input value of specified port

- **Example**

  /* Get the PT6 port input data value */

  uint32_t i32Port; i32Port = DrvGPIO_PT6_GetPortBits();

5.3.72. DrvGPIO_PT6_SetPortBits

- **Prototype**

  void DrvGPIO_PT6_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Configure the register 0x40850[17][1]/ 0x40854[17][1]/ 0x40858[17][1] / 0x4085C[17][1]

- **Parameters**

  i32Data [in] specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* Set PT6.2 ,PT6.4 as 1 */

  DrvGPIO_PT6_SetPortBits(0x14);

5.3.73. DrvGPIO_PT6_ClrPortBits

- **Prototype**

  void DrvGPIO_PT6_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Configure the register 0x40850[17][1]/ 0x40854[17][1]/ 0x40858[17][1] / 0x4085C[17][1]

- **Parameters**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* clear PT6.1, PT6.4 as 0 */

  DrvGPIO_PT6_ClrPortBits(0x12);

5.3.74. DrvGPIO_PT7_EnableINPUT

- **Prototype**

  void DrvGPIO_PT7_EnableINPUT(short int ubit)

- **Description**

  Enable the input mode of the specified GPIO pin .

  Configure the 0x40860[18][2]/ 0x40864[18][2]/ 0x40868[18][2] / 0x4086C[18][2]

- **Parameters**

  ubit[in] : specified PT7pin. It could be 0~0xff

  Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

© 2021-2022HYCON Technology Corp

None

- **Example**

/* set PT7.0/PT7.1 as input mode*/

DrvGPIO_PT7_EnableINPUT(0x01|0x02);

5.3.75. DrvGPIO_PT7_DisableINPUT

- **Prototype**

void DrvGPIO_PT7_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40860[18][2]/ 0x40864[18][2]/ 0x40868[18][2] / 0x4086C[18][2]

- **Parameters**

ubit[in] : specified PT7 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* disable the input mode of PT7.0/PT7.1*/

DrvGPIO_PT7_DisableINPUT(0x01|0x02);

5.3.76. DrvGPIO_PT7_EnableOUTPUT

- **Prototype**

void DrvGPIO_PT7_EnableOUTPUT(short int ubit)

- **Description**

Enable the output mode of the specified GPIO pin .

Configure the register 0x40860[19][3]/ 0x40864[19][3]/ 0x40868[19][3] / 0x4086C[19][3]

- **Parameters**

ubit[in] : specified PT7 pin. It could be 0~0xff

Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

© 2021-2022HYCON Technology Corp

- **Example**

  /* set PT7.0/PT7.1 as output mode*/

  DrvGPIO_PT7_EnableOUTPUT(0x01|0x02);

5.3.77. DrvGPIO_PT7_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT7_DisableOUTPUT(short int ubit)

- **Description**

  Disable the output mode of the specified GPIO pin .

  Configure the register 0x40860[19][3]/ 0x40864[19][3]/0x40868[19][3] / 0x4086C[19][3]

- **Parameters**

  ubit[in] : specified PT7 pin. It could be 0~0xff

  Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* disable the output mode of PT7.0/PT7.1*/

  DrvGPIO_PT7_DisableOUTPUT(0x01|0x02);

5.3.78. DrvGPIO_PT7_GetPortBits

- **Prototype**

  unsigned char DrvGPIO_PT7_GetPortBits (void)

- **Description**

  Get the input port data from the specified GPIO port.

  Read the register 0x40860[16][0]/ 0x40864[16][0]/ 0x40868[16][0] / 0x4086C[16][0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0 ~ 0xFF :The input value of specified port

- **Example**

  /* Get the PT7 port input data value */

  uint32_t i32Port; i32Port = DrvGPIO_PT7_GetPortBits();

© 2021-2022HYCON Technology Corp

5.3.79. DrvGPIO_PT7_SetPortBits

- **Prototype**

  void DrvGPIO_PT7_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Configure the register 0x40860[17][1]/ 0x40864[17][1]/ 0x40868[17][1] / 0x4086C[17][1]

- **Parameters**

  i32Data [in] : specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* Set PT7.2, PT7.4 as 1 */
DrvGPIO_PT7_SetPortBits(0x14);
```

5.3.80. DrvGPIO_PT7_ClrPortBits

- **Prototype**

  void DrvGPIO_PT7_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Configure the register 0x40860[17][1]/ 0x40864[17][1]/ 0x40868[17][1] / 0x4086C[17][1]

- **Parameters**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* clear PT7.1, PT7.4 as 0 */
DrvGPIO_PT7_ClrPortBits(0x12);
```

© 2021-2022HYCON Technology Corp

5.3.81. DrvGPIO_PT8_EnableINPUT

- **Prototype**

  void DrvGPIO_PT8_EnableINPUT(short int ubit)

- **Description**

  Enable the input mode of the specified GPIO pin .

  Configure the register 0x40870[18][2]/ 0x40874[18][2]/ 0x40878[18][2] / 0x4087C[18][2]

- **Parameters**

  ubit[in] : specified PT8pin. It could be 0~0xff

  Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* set PT8.0/PT8.1 as input mode*/

  DrvGPIO_PT8_EnableINPUT(0x01|0x02);

5.3.82. DrvGPIO_PT8_DisableINPUT

- **Prototype**

  void DrvGPIO_PT8_DisableINPUT(short int ubit)

- **Description**

  Disable the input mode of the specified GPIO pin .

  Configure the register 0x40870[18][2]/ 0x40874[18][2]/ 0x40878[18][2] / 0x4087C[18][2]

- **Parameters**

  ubit[in] : specified PT8 pin. It could be 0~0xff

  Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* disable the input mode of PT8.0/PT8.1*/

  DrvGPIO_PT8_DisableINPUT(0x01|0x02);

© 2021-2022HYCON Technology Corp

5.3.83. DrvGPIO_PT8_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT8_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

  Configure the register 0x40870[19][3]/ 0x40874[19][3]/ 0x40878[19][3] / 0x4087C[19][3]

- **Parameters**

  ubit[in] : specified PT8 pin. It could be 0~0xff

  Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* set PT8.0/PT8.1 as output mode*/

  DrvGPIO_PT8_EnableOUTPUT(0x01|0x02);

5.3.84. DrvGPIO_PT8_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT8_DisableOUTPUT(short int ubit)

- **Description**

  Disable the output mode of the specified GPIO pin .

  Configure the register 0x40870[19][3]/ 0x40874[19][3]/ 0x40878[19][3] / 0x4087C[19][3]

- **Parameters**

  ubit[in] : specified PT8 pin. It could be 0~0xff

  Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* disable the output mode of PT8.0/PT8.1*/

  DrvGPIO_PT8_DisableOUTPUT(0x01|0x02);

© 2021-2022HYCON Technology Corp

5.3.85. DrvGPIO_PT8_GetPortBits

- **Prototype**

  unsigned char DrvGPIO_PT8_GetPortBits (void)

- **Description**

  Get the input port data from the specified GPIO port.

  Read the register 0x40870[16][0]/ 0x40874[16][0]/ 0x40878[16][0] / 0x4087C[16][0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  0 ~ 0xFF :The input value of specified port

- **Example**

  /* Get the PT8 port input data value */

  uint32_t i32Port; i32Port = DrvGPIO_PT8_GetPortBits();

5.3.86. DrvGPIO_PT8_SetPortBits

- **Prototype**

  void DrvGPIO_PT8_SetPortBits (unsigned char ui32Data)

- **Description**

  Set the output port value to the specified pin.

  Configure the register 0x40870[17][1]/ 0x40874[17][1]/ 0x40878[17][1] / 0x4087C[17][1]

- **Include**

  i32Data [in] : specify which bit to be set. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* Set PT8.2, PT8.4 as 1 */

  DrvGPIO_PT8_SetPortBits(0x14);

5.3.87. DrvGPIO_PT8_ClrPortBits

- **Prototype**

void DrvGPIO_PT8_ClrPortBits (unsigned int ui32Data)

- **Description**

Clear the output data of the specified pin.

Configure the register 0x40870[17][1]/ 0x40874[17][1]/ 0x40878[17][1] / 0x4087C[17][1]

- **Parameters**

i32Data [in] : specify which bit to be clear. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* clear PT8.1, PT8.4 as 0 */
DrvGPIO_PT8_ClrPortBits(0x12);
```

5.3.88. DrvGPIO_PT9_EnableINPUT

- **Prototype**

void DrvGPIO_PT9_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40880[18][2]/ 0x40884[18][2]/ 0x40888[18][2] / 0x4088C[18][2]

- **Parameters**

ubit[in] : specified PT9pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT9.0/PT9.1 as input mode*/
DrvGPIO_PT9_EnableINPUT(0x01|0x02);
```

5.3.89. DrvGPIO_PT9_DisableINPUT

- **Prototype**

void DrvGPIO_PT9_DisableINPUT(short int ubit)

© 2021-2022HYCON Technology Corp

- **Description**

  Disable the input mode of the specified GPIO pin .

  Configure the register 0x40880[18][2]/ 0x40884[18][2]/ 0x40888[18][2] / 0x4088C[18][2]

- **Parameters**

  ubit[in] : specified PT9 pin. It could be 0~0xff

  Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* disable the input mode of PT9.0/PT9.1*/
DrvGPIO_PT9_DisableINPUT(0x01|0x02);
```

5.3.90. DrvGPIO_PT9_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT9_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

  Configure the register 0x40880[19][3]/ 0x40884[19][3]/ 0x40888[19][3] / 0x4088C[19][3]

- **Parameters**

  ubit[in] : specified PT9 pin. It could be 0~0xff

  Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* set PT9.0/PT9.1 as output mode*/
DrvGPIO_PT9_EnableOUTPUT(0x01|0x02);
```

5.3.91. DrvGPIO_PT9_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT9_DisableOUTPUT(short int ubit)

- **Description**

Disable the output mode of the specified GPIO pin .

Configure the register 0x40880[19][3]/ 0x40884[19][3]/ 0x40888[19][3] / 0x4088C[19][3]

- **Parameters**

ubit[in] : specified PT9 pin. It could be 0~0xff

Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* disable the output mode of PT9.0/PT9.1*/

DrvGPIO_PT9_DisableOUTPUT(0x01|0x02);

5.3.92. DrvGPIO_PT9_GetPortBits

- **Prototype**

unsigned char DrvGPIO_PT9_GetPortBits (void)

- **Description**

Get the input port data from the specified GPIO port.

Read the register 0x40880[16][0]/ 0x40884[16][0]/ 0x40888[16][0] / 0x4088C[16][0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0 ~ 0xFF :The input value of specified port

- **Example**

/* Get the PT9 port input data value */

uint32_t i32Port; i32Port = DrvGPIO_PT9_GetPortBits();

5.3.93. DrvGPIO_PT9_SetPortBits

- **Prototype**

void DrvGPIO_PT9_SetPortBits (unsigned char ui32Data)

- **Description**

Set the output port value to the specified pin.

Configure the register 0x40880[17][1]/ 0x40884[17][1]/ 0x40888[17][1] / 0x4088C[17][1]

- **Parameters**

© 2021-2022HYCON Technology Corp

i32Data [in] : specify which bit to be set. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* Set PT9.2, PT9.4 as 1 */
DrvGPIO_PT9_SetPortBits(0x14);
```

5.3.94. DrvGPIO_PT9_ClrPortBits

- **Prototype**

void DrvGPIO_PT9_ClrPortBits (unsigned int ui32Data)

- **Description**

Clear the output data of the specified pin.

Configure the register 0x40880[17][1]/ 0x40884[17][1]/ 0x40888[17][1] / 0x4088C[17][1]

- **Parameters**

i32Data [in] : specify which bit to be clear. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* clear PT9.1, PT9.4 as 0 */
DrvGPIO_PT9_ClrPortBits(0x12);
```

5.3.95. DrvGPIO_PT10_EnableINPUT

- **Prototype**

void DrvGPIO_PT10_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x40890[18][2]/ 0x40894[18][2]/ 0x40898[18][2] / 0x4089C[18][2]

- **Parameters**

ubit[in] : specified PT10pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT10.0/PT10.1 as input mode*/
DrvGPIO_PT10_EnableINPUT(0x01|0x02);
```

5.3.96. DrvGPIO_PT10_DisableINPUT

- **Prototype**

void DrvGPIO_PT10_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x40890[18][2]/ 0x40894[18][2]/ 0x40898[18][2] / 0x4089C[18][2]

- **Parameters**

ubit[in] : specified PT10 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* disable the input mode of PT10.0/PT10.1*/
DrvGPIO_PT10_DisableINPUT(0x01|0x02);
```

5.3.97. DrvGPIO_PT10_EnableOUTPUT

- **Prototype**

void DrvGPIO_PT10_EnableOUTPUT(short int ubit)

- **Description**

Enable the output mode of the specified GPIO pin .

Configure the register 0x40890[19][3]/ 0x40894[19][3]/ 0x40898[19][3] / 0x4089C[19][3]

- **Parameters**

ubit[in] : specified PT10 pin. It could be 0~0xff

© 2021-2022HYCON Technology Corp

Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* set PT10.0/PT10.1 as output mode*/

DrvGPIO_PT10_EnableOUTPUT(0x01|0x02);

5.3.98. DrvGPIO_PT10_DisableOUTPUT

- **Prototype**

void DrvGPIO_PT10_DisableOUTPUT(short int ubit)

- **Description**

Disable the output mode of the specified GPIO pin .

Configure the register 0x40890[19][3]/ 0x40894[19][3]/ 0x40898[19][3] / 0x4089C[19][3]

- **Parameters**

ubit[in] : specified PT10 pin. It could be 0~0xff

Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

/* disable the output mode of PT10.0/PT10.1*/

DrvGPIO_PT10_DisableOUTPUT(0x01|0x02);

5.3.99. DrvGPIO_PT10_GetPortBits

- **Prototype**

unsigned char DrvGPIO_PT10_GetPortBits (void)

- **Description**

Get the input port data from the specified GPIO port.

Read the register 0x40890[16][0]/ 0x40894[16][0]/ 0x40898[16][0] / 0x4089C[16][0]

- **Parameters**

None

- **Include**

© 2021-2022HYCON Technology Corp

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0 ~ 0xFF :The input value of specified port

- **Example**

```
/* Get the PT10 port input data value */
uint32_t    i32Port; i32Port = DrvGPIO_PT10_GetPortBits();
```

5.3.100. DrvGPIO_PT10_SetPortBits

- **Prototype**

void DrvGPIO_PT10_SetPortBits (unsigned char ui32Data)

- **Description**

Set the output port value to the specified pin.

Configure the register 0x40890[17][1]/ 0x40894[17][1]/ 0x40898[17][1] / 0x4089C[17][1]

- **Parameters**

i32Data [in] specify which bit to be set. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* Set PT10.1  PT10.4 */
DrvGPIO_PT10_SetPortBits(0x01|0x02);
```

5.3.101. DrvGPIO_PT10_ClrPortBits

- **Prototype**

void DrvGPIO_PT10_ClrPortBits (unsigned int ui32Data)

- **Description**

Clear the output data of the specified pin.

Configure the register 0x40890[17][1]/ 0x40894[17][1]/ 0x40898[17][1] / 0x4089C[17][1]

- **Parameters**

i32Data [in] : specify which bit to be clear. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

© 2021-2022HYCON Technology Corp

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* clear PT10.1    PT10.0 */
DrvGPIO_PT10_ClrPortBits(0x1|0x2);
```

5.3.102. DrvGPIO_PT13_EnableINPUT

- **Prototype**

void DrvGPIO_PT13_EnableINPUT(short int ubit)

- **Description**

Enable the input mode of the specified GPIO pin .

Configure the register 0x408C0[18][2]/ 0x408C4[18][2]/ 0x408C8[18][2] / 0x408CC[18][2]

- **Parameters**

ubit[in] : specified PT13 pin. It could be 0~0xff

Set the specified GPIO pin to the input operation mode if the bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

```
/* set PT13.0/PT13.1 as input mode*/
DrvGPIO_PT13_EnableINPUT(0x01|0x02);
```

5.3.103. DrvGPIO_PT13_DisableINPUT

- **Prototype**

void DrvGPIO_PT13_DisableINPUT(short int ubit)

- **Description**

Disable the input mode of the specified GPIO pin .

Configure the register 0x408C0[18][2]/ 0x408C4[18][2]/ 0x408C8[18][2] / 0x408CC[18][2]

- **Parameters**

ubit[in] : specified PT13 pin. It could be 0~0xff

Disable the input mode of the specified GPIO pin if the specified bit of ubit is equal to 1

The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

Peripheral_lib/DrvGPIO.h

© 2021-2022HYCON Technology Corp

- **Return Vaule**

  None

- **Example**

  /* disable the input mode of PT13.0/PT13.1*/

  DrvGPIO_PT13_DisableINPUT(0x01|0x02);

5.3.104. DrvGPIO_PT13_EnableOUTPUT

- **Prototype**

  void DrvGPIO_PT13_EnableOUTPUT(short int ubit)

- **Description**

  Enable the output mode of the specified GPIO pin .

  Configure the register 0x408C0[19][3]/ 0x408C4[19][3]/ 0x408C8[19][3] / 0x408CC[19][3]

- **Parameters**

  ubit[in] : specified PT13 pin. It could be 0~0xff

  Set the specified GPIO pin to the output operation mode if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

  /* set PT13.0/PT13.1 as output mode*/

  DrvGPIO_PT13_EnableOUTPUT(0x01|0x02);

5.3.105. DrvGPIO_PT13_DisableOUTPUT

- **Prototype**

  void DrvGPIO_PT13_DisableOUTPUT(short int ubit)

- **Description**

  Disable the output mode of the specified GPIO pin .

  Configure the register 0x408C0[19][3]/ 0x408C4[19][3]/ 0x408C8[19][3] / 0x408CC[19][3]

- **Parameters**

  ubit[in] : specified PT13 pin. It could be 0~0xff

  Disable the output mode of the specified GPIO pin if the specified bit of ubit is equal to 1

  The operation mode of specified GPIO pin would not change if the specified bit of ubit is equal to 0

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

© 2021-2022HYCON Technology Corp

None

- **Example**

/* disable the output mode of PT13.0/PT13.1*/

DrvGPIO_PT13_DisableOUTPUT(0x01|0x02);

5.3.106. DrvGPIO_PT13_GetPortBits

- **Prototype**

unsigned char DrvGPIO_PT13_GetPortBits (void)

- **Description**

Get the input port data from the specified GPIO port.

Read the register 0x408C0[16][0]/ 0x408C4[16][0]/ 0x408C8[16][0] / 0x408CC[16][0]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

0 ~ 0xFF :The input value of specified port

- **Example**

/* Get the PT13 port input data value */

uint32_t i32Port; i32Port = DrvGPIO_PT13_GetPortBits();

5.3.107. DrvGPIO_PT13_SetPortBits

- **Prototype**

void DrvGPIO_PT13_SetPortBits (unsigned char ui32Data)

- **Description**

Set the output port value to the specified pin.

Configure the register 0x408C0[17][1]/ 0x408C4[17][1]/ 0x408C8[17][1] / 0x408CC[17][1]

- **Parameters**

i32Data [in] specify which bit to be set. It could be 0~0xFF

The each bit of i32Data corresponding to one pin .

Output data of the specified GPIO pin will be set 1 when the bit of u32Bit is equal to 1, the bit will be set 0 if the bit of the i32Data is equal to 0.

- **Include**

Peripheral_lib/DrvGPIO.h

- **Return Vaule**

None

- **Example**

© 2021-2022HYCON Technology Corp

```
/* Set PT13.1    PT13.4 */
DrvGPIO_PT13_SetPortBits(0x01|0x02);
```

5.3.108. DrvGPIO_PT13_ClrPortBits

- **Prototype**

  void DrvGPIO_PT13_ClrPortBits (unsigned int ui32Data)

- **Description**

  Clear the output data of the specified pin.

  Configure the register 0x408C0[17][1]/ 0x408C4[17][1]/ 0x408C8[17][1] / 0x408CC[17][1]

- **Parameters**

  i32Data [in] : specify which bit to be clear. It could be 0~0xFF

  The each bit of i32Data corresponding to one pin .

  Output data of the specified GPIO pin will be clear when the corresponding bit of u32Bit is equal to 1

- **Include**

  Peripheral_lib/DrvGPIO.h

- **Return Vaule**

  None

- **Example**

```
/* clear PT13.1    PT13.0 */
DrvGPIO_PT13_ClrPortBits(0x1|0x2);
```

© 2021-2022HYCON Technology Corp

6. ADC Driver

6.1. Introduction

The following functions are included in ADC Manager Section.

Item	Functions	Description
01	DrvADC_PInputChannel	Positive input source
02	DrvADC_NInputChannel	Negative input source
03	DrvADC_SetADCInputChannel	Set the ADC input mode
04	DrvADC_InputSwitch	ADC input short control
05	DrvADC_RefInputShort	ADC reference short control
06	DrvADC_SetPGA	Input signal gain
07	DrvADC_ADGain	Input signal gain
08	DrvADC_Gain	Input signal gain
09	DrvADC_DCoffset	DC offset input selection
10	DrvADC_RefVoltage	Set the ADC reference
11	DrvADC_FullRefRange	Set the ADC reference
12	DrvADC_OSR	Set the ADC OSR
13	DrvADC_ACM	ACM input source
14	DrvADC_ClkEnable	Enable ADC clock
15	DrvADC_ClkDisable	Disable ADC clock
16	DrvADC_CombFilter	Comb filter enable control
17	DrvADC_EnableInt	ADC Interrupt Enable
18	DrvADC_DisableInt	ADC Interrupt Disable
19	DrvADC_ReadIntFlag	Read ADC interrupt flag
20	DrvADC_ClearIntFlag	Clear ADC interrupt flag
21	DrvADC_Enable	Enable ADC control
22	DrvADC_Disable	Disable ADC control
23	DrvADC_GetConversionData	Get the A/D conversion data

© 2021-2022HYCON Technology Corp

6.2. Type Definition

E_ADC_INPUT_CHANNEL

Enumeration Identifier	Value	Description
ADC_Input_AIO0	0	Signal input
ADC_Input_AIO1	1	Signal input
ADC_Input_AIO2	2	Signal input
ADC_Input_AIO3	3	Signal input
REFO_I	4	Signal input
VDD5VD10	5	Signal input
VSS_INN	5	Signal input
TSP0	6	Signal input
TSP1	7	Signal input
VDDA_IN	8	Signal input
ADC_Input_AIO4	9	Signal input
ADC_Input_AIO5	10	Signal input
ADC_Input_AIO6	11	Signal input
ADC_Input_AIO7	12	Signal input
ADC_Input_AIO8	13	Signal input
VSS_INP	14	Signal input

E_ADC_REFV

Enumeration Identifier	Value	Description
External	0	External
Internal	1	Enable buffer and use internal source

E_ADC_PGA & E_ADC_ADGN

Enumeration Identifier	Value	Description	Enumeration Identifier	Value	Description
ADC_PGA_Disable	0	Disable PGA	ADC_ADGN_1	0	ADGN=1
ADC_PGA_8	1	PGA=8	ADC_ADGN_2	1	ADGN=2
ADC_PGA_16	3	PGA=16	ADC_ADGN_RESER	2	Reserve
ADC_PGA_32	7	PGA=32	ADC_ADGN_4	3	ADGN=4

E_ADC_SIGNAL_SHORT

Enumeration Identifier	Value	Description
OPEN	0	ADC signal input (positive and negative)open control
SHORT	1	ADC signal input(positive and negative)short control

E_ADC_VRPS_REF_VOLTAGE

Enumeration Identifier	Value	Description
VDDA	0	Reference voltage VDDA
AIO2	1	Reference voltage form AIO2
AIO4	2	Reference voltage form AIO4
REF_BUFFER_OUT	3	Reference voltage form REFO_I

E_ADC_VRNS_REF_VOLTAGE

Enumeration Identifier	Value	Description
VSSA	0	Reference voltage VSSA
AIO3	1	Reference voltage form AIO3

© 2021-2022HYCON Technology Corp

AIO5	2	Reference voltage form AIO5
REF_BUFFER_OUT	3	Reference voltage form REFO_I

6.3. Functions

6.3.1. DrvADC_PInputChannel

- **Prototype**

unsigned int DrvADC_PInputChannel (E_ADC_INPUT_Channel uINP);

- **Description**

Set the ADC positive input voltage source.

Configure the register 0x41104[7:4].

- **Parameters**

uINP [in] : Specify the input channel. It could be 0~13

0：AIO0, 1：AIO1,

2：AIO2, 3：AIO3,

4：REFO_I, 5：VDD5VD10,

6：TSP0, 7：TSP1,

8： VDDA_IN, 9：AIO4;

10：AIO5, 11：AIO6

12：AIO7, 13：AIO8

14：VSS_INP

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Select the positive input voltage source form AIO0*/
DrvADC_PInputChannel(ADC_Input_AIO0);
```

6.3.2. DrvADC_NInputChannel

- **Prototype**

unsigned int DrvADC_NInputChannel (E_ADC_INPUT_Channel uINN);

- **Description**

Set the ADC negative input voltage source.

Configure the register 0x41104[3:0].

- **Parameters**

uINN [in] : Specify the input channel. It could be 0~13

0：AIO0, 1：AIO1,

2：AIO2, 3：AIO3,

4：REFO_I,　5：VSS,

6：TSN0,　7：TSN1,

8：VDDA_IN,　9：AIO4

10：AIO5,　11：AIO6

12：AIO7,　13：AIO8

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Select the negative input voltage source form AIO1*/
DrvADC_NInputChannel(ADC_Input_AIO1);
```

6.3.3. DrvADC_SetADCInputChannel

- **Prototype**

```
unsigned int DrvADC_SetADCInputChannel (E_ADC_INPUT_Channel uINP,
                                        E_ADC_INPUT_Channel uINN  );
```

- **Description**

Set the ADC input source.

Configure the register 0x41104[7:4] / 0x41104[3:0].

- **Parameters**

uINP [in] : Specify the positive input channel. It could be 0~13

0：AIO0,　1：AIO1,

2：AIO2,　3：AIO3,

4：REFO_I,　5：VDD5VD10,

6：TSP0,　7：TSP1,

8：　VDDA_IN,　9：AIO4;

10：AIO5,　11：AIO6

12：AIO7,　13：AIO8

14：VSS_INP

uINN [in] : Specify the negative input channel. It could be 0~13

0：AIO0,　1：AIO1,

2：AIO2,　3：AIO3,

4：REFO_I,　5：VSS,

6：TSN0,　7：TSN1,

8：VDDA_IN,　9：AIO4

10：AIO5,　11：AIO6

© 2021-2022HYCON Technology Corp

12：AIO7,　13：AIO8

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* the following statement indicates that the external analog input is AIO0 and AIO1 input */
DrvADC_SetADCInputChannel(ADC_Input_AIO0, ADC_Input_AIO1);
```

6.3.4. DrvADC_InputSwitch

- **Prototype**

  unsigned int DrvADC_InputSwitch (uVISHR)

- **Description**

  ADC signal input (positive and negative) short control.

  Configure the register 0x41100[21]

- **Parameters**

  uVISHR[in] : ADC input short switch.

  0: OPEN

  1: SHORT

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* ADC input short */
  DrvADC_InputSwitch(1);
```

6.3.5. DrvADC_RefInputShort

- **Prototype**

  unsigned int DrvADC_RefInputShort (E_ADC_SIGNAL_SHORT uVrshr);

- **Description**

  Set the ADC reference input (positive and negative) short control.

  Configure the register 0x41100[20].

- **Parameters**

  uVrshr [in] : ADC reference input short control.

0 : ADC reference input (positive and negative)open control

1 : ADC reference input(positive and negative)short control

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the ADC reference input short */
DrvADC_RefInputShort(SHORT);
```

6.3.6. DrvADC_SetPGA

- **Prototype**

unsigned int DrvADC_SetPGA (E_ADC_PGA uPGA);

- **Description**

Input signal gain for ADC modulator.

Configure the register 0x41104[18:16].

- **Parameters**

uPGA [in] : Specify the ADC PGA.

0: Gain=1

1: Gain=8

2: Reserved

3: Gain=16

4: Reserved

5: Reserved

6: Reserved

7: Gain=32

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the gain of 8 */
DrvADC_SetPGA(ADC_Gain_8);
```

6.3.7. DrvADC_ADGain

- **Prototype**

  unsigned int DrvADC_ADGain (uADgain);

- **Description**

  Input signal gain for ADC modulator.

  Configure the register 0x41104[21:20]

- **Parameters**

  uADgain [in] : Specify the ADC ADGN.

  0: Gain=1

  1: Gain=2

  3: Gain=4

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Set the gain of 2 */

  DrvADC_ADGain(1);

6.3.8. DrvADC_Gain

- **Prototype**

  unsigned int DrvADC_Gain (E_ADC_PGA uPGA ,uADgain);

- **Description**

  Input signal gain for modulator.

  Configure the register 0x41104[18:16]/ 0x41104[21:20]

- **Parameters**

  uPGA [in] : Specify the ADC PGA.

  0: Gain=1

  1: Gain=8

  2: Reserved

  3: Gain=16

  4: Reserved

  5: Reserved

  6: Reserved

  7: Gain=32

  uADgain [in] : Specify the ADC ADGN.

  0: Gain=1

  1: Gain=2

© 2021-2022HYCON Technology Corp

3: Gain=4

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Set the total gain of 128 */

DrvADC_Gain(7,3);

6.3.9. DrvADC_DCoffset

- Prototype

unsigned int DrvADC_DCoffset (uDCoffset);

- **Description**

DC offset input voltage selection (VREF=REFP-REFN)

Configure the register 0x41104[27:24]

- **Parameters**

uDCoffice [in] : Specify the ADC DCSET.

0 : 0 VREF

1 : +1/8 VREF

2 : +1/4 VREF

3 : +3/8 VREF

4 : +1/2 VREF

5 : +5/8 VREF

6 : +3/4 VREF

7 : +7/8 VREF

8 : 0 VREF

9 : -1/8 VREF

10 : -1/4 VREF

11 : -3/8 VREF

12 : -1/2 VREF

13 : -5/8 VREF

14 : -3/4 VREF

15 : -7/8 VREF

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

© 2021-2022HYCON Technology Corp

Other : Incorrect argument

- **Example**

/* Set the DC offset of +1/8 voltage. */

DrvADC_DCoffset(1);

6.3.10. DrvADC_RefVoltage

- **Prototype**

unsigned int DrvADC_RefVoltage (E_ADC_VRPS_REF_VOLTAGE uVrps,
E_ADC_VRNS_REF_VOLTAGE uVrns);

- **Description**

Set the ADC reference voltage.

Configure the register 0x41100[19:18] and 0x41100[17:16].

- **Parameters**

uVrps [in] : Specify the ADC VRPS, the input range is 0~3

0 : Reference voltage VDDA

1 : Reference voltage form AIO2

2 : Reference voltage form AIO4

3 : Reference voltage form REFO_I

uVrns [in] : Specify the ADC VRNS, the input range is 0~3

0 : Reference voltage VSSA

1 : Reference voltage form AIO3

2 : Reference voltage form AIO5

3 : Reference voltage form REFO_I

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Set the ADC reference voltage.(VRPS=AIO2, VRNS=AIO3) */

DrvADC_RefVoltage(AIO2, AIO3);

6.3.11. DrvADC_FullRefRange

- **Prototype**

unsigned int DrvADC_FullRefRange(uFullRange);

- **Description**

Set the ADC full reference range select.

Configure the register 0x41104[19]

- **Parameters**

uFullRange [in] : Specify the VREF gain. VREF= VRPS-VRNS

0: Full reference range input=VREF*1

1: 1/2 reference range input=VREF*1/2

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the ADC full reference range input. */
DrvADC_FullRefRange(0);
```

6.3.12. DrvADC_OSR

- **Prototype**

unsigned int DrvADC_OSR (uADCOSR);

- **Description**

Set the ADC OSR. Configure the register 0x41100[5:2]

- **Parameters**

uADCOSR [in] : Specify the ADC OSR. (The following output rate is calculated when clock is 327680HZ)

0 : ÷32768，Data Output Rate is10sps

1 : ÷16384，Data Output Rate is 20sps

2 : ÷8192，Data Output Rate is 40sps

3 : ÷4096，Data Output Rate is 80sps

4 : ÷2048，Data Output Rate is 160sps

5 : ÷1024，Data Output Rate is 320sps

6 : ÷512，Data Output Rate is 640sps

7 : ÷256，Data Output Rate is 1280sps

8 : ÷128 ，Data Output Rate is 2560sps

9 : ÷64，Data Output Rate is 5120sps

10 : ÷32，Data Output Rate is 10240sps

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

© 2021-2022HYCON Technology Corp

```
/* Set the OSR of 8192 data rate 40sps. */
DrvADC_OSR(2);
```

6.3.13. DrvADC_ACM

- **Prototype**

```
unsigned int DrvADC_ACM(uACMS);
```

- **Description**

Set the ACM input source

Configure the register 0x41100[7].

- **Parameters**

uACMS [in] : Specify the ACM input source.

0 : ACM_REFO_I

1 : V12

- **Include**

Peripheral_lib/DrvADC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the ACM input source 1.2V */
DrvADC_ACM(V12);
```

6.3.14. DrvADC_ClkEnable

- **Prototype**

```
unsigned int DrvADC_ClkEnable(uADCD);
```

- **Description**

Enable ADC clock, set the clock divider, the ADC clock phase adjustment

Configure the register 0x4030C[7:4].

- **Parameters**

uADCD [in] : Specify the ADC clock divider.

0 : ÷6

1 : ÷12

2 : ÷30

3 : ÷60

uClkPH [in] :

0 : ADC clock rising edge of CPU clock low.

1 : ADC clock rising edge of CPU clock high.

- **Include**

© 2021-2022HYCON Technology Corp

Peripheral_lib/DrvADC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Set the ADCD of ÷12 , ADC clock rising edge of CPU clock high. */
DrvADC_ClkEnable(1,1);
```

6.3.15. DrvADC_ClkDisable

- **Prototype**

```
void DrvADC_ClkDisable(void);
```

- **Description**

  Disable ADC clock.

  Configure the register 0x4030C[6]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Disable ADC clock. */
DrvADC_ClkDisable();
```

6.3.16. DrvADC_CombFilter

- **Prototype**

```
unsigned int DrvADC_CombFilter(uCFRST);
```

- **Description**

  Comb filter enable control

  Configure the register 0x41100[1]

- **Parameters**

  uCFRST [in] : Comb filter enable control

  0: Reset

  1: On

- **Include**

  Peripheral_lib/DrvADC.h

© 2021-2022HYCON Technology Corp

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Open the comb filter. */
DrvADC_CombFilter(0);// reset the comb filter
DrvADC_CombFilter(1);//enable the comb filter
```

6.3.17. DrvADC_EnableInt

- **Prototype**

  void DrvADC_EnableInt (void)

- **Description**

  ADC Interrupt Enable

  Configure the register 0x40008[16]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  None

- **Example**

  ADC Interrupt Enable

  Configure the register 0x40008[16]=1b

6.3.18. DrvADC_DisableInt

- **Prototype**

  void DrvADC_DisableInt (void)

- **Description**

  ADC Interrupt Disable

  Configure the register 0x40008[16]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  None

- **Example**

© 2021-2022HYCON Technology Corp

```
/* Disable ADC interrupt */
DrvADC_DisableInt();
```

6.3.19. DrvADC_ReadIntFlag

- **Prototype**

  unsigned int DrvADC_ReadIntFlag (void)

- **Description**

  Read ADC interrupt flag.

  Read the register 0x40008[0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  0 : Interrupt flag is 0, No interrupt occurred.

  1 : Interrupt flag is 1, interrupt occurred.

  >1: Invalid return value

- **Example**

```
/* Read ADC interrupt flag */
flag=DrvADC_ReadIntFlag();
```

6.3.20. DrvADC_ClearIntFlag

- **Prototype**

  void DrvADC_ClearIntFlag (void)

- **Description**

  Clear ADC interrupt flag.

  Configure the register 0x40008[0]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  None

- **Example**

```
/* Clear ADC interrupt flag */
DrvADC_ClearIntFlag();
```

© 2021-2022HYCON Technology Corp

6.3.21. DrvADC_Enable

- **Prototype**

  void DrvADC_Enable(void)

- **Description**

  Enable ADC control

  Configure the register 0x41100[0]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  None

- **Example**

```
/* Enable ADC */
DrvADC_Enable();
```

6.3.22. DrvADC_Disable

- **Prototype**

  void DrvADC_Disable(void)

- **Description**

  Disable ADC control. Configure the register 0x41100[0]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  None

- **Example**

```
/* Disable ADC */
DrvADC_Disable();
```

6.3.23. DrvADC_GetConversionData

- **Prototype**

  int DrvADC_GetConversionData (void);

- **Description**

  Get the A/D conversion data with signed.

  Configure the register 0x41108[31:0]

© 2021-2022HYCON Technology Corp

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvADC.h

- **Return Vaule**

  Return the conversion data.

- **Example**

```
/* Get the ADC conversion data */
int adc_data ;
adc_data=DrvADC_GetConversionDate();
```

7. SPI32 Driver

7.1. Introduction

The following functions are included in SPI Manager Section.

Item	Functions	Description
01	DrvSPI32_Open	Open SPI module
02	DrvSPI32_Close	Close SPI module
03	DrvSPI32_IsBusy	Check busy status
04	DrvSPI32_CLKSource	Configure the frequency of SPI clock
05	DrvSPI32_IsRxBufferFull	Check Rx buffer status
06	DrvSPI32_IsTxBufferFull	Check Tx buffer status
07	DrvSPI32_EnableRxInt	Enable the SPI Rx interrupt
08	DrvSPI32_EnableTxInt	Enable the SPI Tx interrupt
09	DrvSPI32_DisableRxInt	Disable the SPI Rx interrupt
10	DrvSPI32_DisableTxInt	Disable the SPI Tx interrupt
11	DrvSPI32_GetRxIntFlag	Get the SPI32 Rx interrupt flag
12	DrvSPI32_GetTxIntFlag	Get the SPI32 Tx interrupt flag
13	DrvSPI32_ClrIntRxFlag	Clear the SPI Rx interrupt flag
14	DrvSPI32_ClrIntTxFlag	Clear the SPI Tx interrupt flag
15	DrvSPI32_Read	Read data from SPIBUF registers
16	DrvSPI32_Write	Write data to SPIBUF register
17	DrvSPI32_Enable	Enable the SPI
18	DrvSPI32_BitLength	Set the SPI transfer Bit length
19	DrvSPI32_GetDCFlag	Get data loss flag of state
20	DrvSPI32_IsABFlag	Read the flag of received data deficient
21	DrvSPI32_IsOVFlag	Read the flag of SPI Bus Data too long
22	DrvSPI32_IsRxFlag	Read the flag of Rx Buffer Updata
23	DrvSPI32_SetEndian	Set the data transmitted from the MSB or LSB start
24	DrvSPI32_SetCSO	Configure CS Polarity
25	DrvSPI32_DisableIO	Disable the SPI port to transmit
26	DrvSPI32_EnableIO	Enable and specify the SPI port to transmit

7.2. Type Definition

E_DRVSPI_MODE

Enumeration Identifier	Value	Description
E_DRVSPI_MASTER1	0	Master,4wire mode
E_DRVSPI_MASTER2	1	Master,3wire mode
E_DRVSPI_MASTER3	2	Master,TI mode
E_DRVSPI_SLAVE1	3	Slave,4wire mode
E_DRVSPI_SLAVE2	4	Slave,3wire mode
E_DRVSPI_SLAVE3	5	Slave,TI mode

E_DRVSPI_TRANS_TYPE

Enumeration Identifier	Value	Description
E_DRVSPI_TYPE0	0	SPI transfer type 0

© 2021-2022HYCON Technology Corp

E_DRVSPI_TYPE1	1	SPI transfer type 1
E_DRVSPI_TYPE2	2	SPI transfer type 2
E_DRVSPI_TYPE3	3	SPI transfer type 3

E_DRVSPI_ENDIAN

Enumeration Identifier	Value	Description
E_DRVSPI_LSB_FIRST	1	Send LSB first
E_DRVSPI_MSB_FIRST	0	Send MSB first

E_DRVSPI_CS

Enumeration Identifier	Value	Description
E_DRVSPI_CSLow	0	CSO low
E_DRVSPI_CSHigh	1	CSO high

© 2021-2022HYCON Technology Corp

7.3. Functions

7.3.1. DrvSPI32_Open

- **Prototype**

unsigned int DrvSPI32_Open(

E_DRVSPI_MODE uMode,

E_DRVSPI_TRANS_TYPE uType,

uOuputPin,

uClkDiv);

- **Description**

This function is used to open SPI module. It decides the SPI to work in master or slave mode, SPI bus timing, specified I / O port. Configure the register

0x4030C[2:0],0x4030C[3]=1b, 0x40844[4]=1b, 0x40844[7:5],0x40F00[3:0],0x40f04[16:17]

uMode : 0x40f00[0]=1b, 0x40f00[1]=xb, 0x40f04[16:17]=0xb. uMode : 0~5

uType : 0x40f00[3:2]=xxb. uType : 0~3

uOuputPin : 0x40844[4]=1b, 0x40844[7:5]=xxxb. uOuputPin : 0~7

uClkDiv : 0x4030C[2:0]=xxxb, 0x4030C[3]=1b. uClkDiv : 0~7

- **Parameters**

uMode [in] : Specify the operation mode

0 : Work in master mode interface 4-wire.

1 : Work in master mode interface 3-wire.

2 : Work in master mode interface TI mode.

3 : Work in slave mode interface 4-wire.

4 : Work in slave mode interface 3-wire.

5 : Work in slave mode interface TI mode.

uType [in] : Transfer types, i.e. the bus timing. It could be 0~ 3.

0: Latch data on first edge of serial clock, clock idle state is low.(CPHA=0 CPOL=0)

1: Latch data on first edge of serial clock, clock idle state is high.(CPHA=0 CPOL=1)

2: Latch data on second edge of serial clock, clock idle state is low.(CPHA=1 CPOL=0)

3: Latch data on second edge of serial clock, clock idle state is high.(CPHA=1 CPOL=1)

uOuputPin [in] : Specify the trasmission port

0 : Port1.0 =CS, Port1.1 =CK, Port1.2 = DI, Port1.3 =DO

1 : Port1.4 =CS, Port1.5 =CK, Port1.6 = DI, Port1.7 =DO

2 : Port2.0 =CS, Port2.1 =CK, Port2.2 = DI, Port2.3 =DO

3 : Port2.4 =CS, Port2.5 =CK, Port2.6 = DI, Port2.7 =DO

4 : Port8.0 =CS, Port8.1 =CK, Port8.2 = DI, Port8.3 =DO

5 : Port8.4 =CS, Port8.5 =CK, Port8.6 = DI, Port8.7 =DO

6 : Port9.0 =CS, Port9.1 =CK, Port9.2 = DI, Port9.3 =DO

© 2021-2022HYCON Technology Corp

7 : Port9.4 =CS, Port9.5 =CK, Port9.6 = DI, Port9.7 =DO

uClkDiv [in] : Specify the clock divider

0 : ÷1

1 : ÷2

2 : ÷4

3 : ÷8

4 : ÷32

5 : ÷128

6 : ÷512

7 : ÷2048

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/*Configure SPI as a master, SPI transfer type 1, Output Pin to select 1: Port2.0 =CS, Port2.1 =CK, Port2.2 = DI, Port2.3 =DO,Set SPI clock/512 */
DrvSPI32_Open(E_DRVSPI_MASTER1, E_DRVSPI_TYPE1, 2,6);
```

7.3.2. DrvSPI32_Close

- **Prototype**

void DrvSPI32_Close (void);

- **Description**

Disable the SPI clock source divider and SPI function and SPI IO port.

Configure the register 0x40F00[0]=0, 0x4030C[3]=0,0x40844[4]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* Close the (SPI) */
DrvSPI32_Close();
```

7.3.3. DrvSPI32_IsBusy

● Prototype

unsigned int DrvSPI32_IsBusy(void);

● Description

Check the busy status of the SPI port.

● Parameters

None

● Include

Peripheral_lib/DrvSPI32.h

● Return Vaule

1: The SPI port is in busy.

0: The SPI port is not in busy.

● Example

```
/* Check the busy status */
unsigned char    flag;
flag=DrvSPI32_IsBusy (); //read 0x40f00[19]
```

7.3.4. DrvSPI32_CLKSource

● **Prototype**

unsigned int DrvSPI32_ CLKSource(uclk);

● **Description**

Configure the frequency of SPI clock. In master mode, the output frequency of serial clock is programmable.

Configure the register 0x40308[1], 0x4030C[2:0]

● **Parameters**

uclk [in] : Specify the SPI clock source

0 : HSXT

1 : HSRC

● **Include**

Peripheral_lib/DrvSPI32.h

● **Return Vaule**

None

● **Example**

```
/* Set SPI clock source HSRC */
DrvSPI32_CLKSource (1);
```

7.3.5. DrvSPI32_IsRxBufferFull

● **Prototype**

unsigned int DrvSPI32_IsRxBufferFull(void);

- **Description**

Check Rx buffer status (only for data reception), read the register 0x40F00[16]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

1: Rx buffer is full.

0: Rx buffer is not full.

- **Example**

```
/* Check the status of Rx buffer */
unsigned char    flag;
flag = DrvSPI32_IsRxBufferFull() ;
```

7.3.6. DrvSPI32_IsTxBufferFull

- **Prototype**

unsigned int DrvSPI32_IsTxBufferFull(void);

- **Description**

Check Tx buffer status

Configure the register 0x40F00[17]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

1: Tx buffer is full.

0: Tx buffer is not full, Tx buffer is empty.

- **Example**

```
/* Check the status of Tx buffer */
unsigned char    flag;    flag =DrvSPI32_IsTxBufferFull();
```

7.3.7. DrvSPI32_EnableRxInt

- **Prototype**

void DrvSPI32_EnableRxInt(void);

- **Description**

Enable the SPI Rx interrupt.

© 2021-2022HYCON Technology Corp

Configure the register 0x40000[16]=1b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* Enable the SPI Rx interrupt */
DrvSPI32_EnableRxInt();
```

7.3.8. DrvSPI32_EnableTxInt

- **Prototype**

void DrvSPI32_EnableTxInt(void);

- **Description**

Enable the SPI Tx interrupt.

Configure the register 0x40000[17]=1b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* Enable the SPI Tx interrupt */
DrvSPI32_EnableTxInt();
```

7.3.9. DrvSPI32_DisableRxInt

- **Prototype**

void DrvSPI32_DisableRxInt(void);

- **Description**

Disable the SPI Rx interrupt.

Configure the register 0x40000[16]=0b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

© 2021-2022HYCON Technology Corp

- **Return Vaule**

  None

- **Example**

  /* Disable the SPI Rx interrupt */

  DrvSPI32_DisableRxInt();

7.3.10. DrvSPI32_DisableTxInt

- **Prototype**

  void DrvSPI32_DisableTxInt(void);

- **Description**

  Disable the SPI Tx interrupt.

  Configure the register 0x40000[17]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  None

- **Example**

  /* Disable the SPI Tx interrupt */

  DrvSPI32_DisableTxInt();

7.3.11. DrvSPI32_GetRxIntFlag

- **Prototype**

  unsigned int DrvSPI32_GetRxIntFlag ();

- **Description**

  Get the SPI RX interrupt flag.

  Read the register 0x40000[0].

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  1: Interrupted

  0: Normal

- **Example**

  /* Get the SPI RX interrupt flag. */

© 2021-2022HYCON Technology Corp

unsigned char flag; flag=DrvSPI_GetRxIntFlag();

7.3.12. DrvSPI32_GetTxIntFlag

- **Prototype**

  unsigned int DrvSPI32_GetTxIntFlag ();

- **Description**

  Get the SPI TX interrupt flag.

  Read the register 0x40000[1]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  0: Interrupted

  1: Normal

- **Example**

  /* Get the SPI Tx interrupt flag. */

  unsigned char flag；

  flag=DrvSPI32_GetTxIntFlag();

7.3.13. DrvSPI32_ClrIntRxFlag

- **Prototype**

  void DrvSPI32_ClrIntRxFlag ();

- **Description**

  Clear the SPI Rx interrupt flag.

  Configure the register 0x40000[0]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  None

- **Example**

  /* Clear the SPI Rx interrupt flag. */

  DrvSPI32_ClrIntRxFlag();

7.3.14. DrvSPI32_ClrIntTxFlag

© 2021-2022HYCON Technology Corp

- **Prototype**

  void DrvSPI32_ClrIntTxFlag ();

- **Description**

  Clear the SPI Tx interrupt flag.

  Configure the register 0x40000[1]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  None

- **Example**

  /* Clear the SPI Tx interrupt flag. */

  DrvSPI32_ClrIntTxFlag();

7.3.15. DrvSPI32_Read

- **Prototype**

  unsigned int DrvSPI32_Read();

- **Description**

  Read data from SPI Rx buffer registers.

  Read the register 0x40F08[31:0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  The return value is SPI Rx buffer register data.

- **Example**

  /*Data transmission：LSB First，8bit*/

  unsigned int data；data=DrvSPI32_Read()>>24;

  /*Data transmission：MSB First，8bit*/

  unsigned int data；data=DrvSPI32_Read();

7.3.16. DrvSPI32_Write

- **Prototype**

  void DrvSPI32_Write (unsigned int uData);

- **Description**

Write data to SPI Tx buffer register. Configure the register 0x40F0C[31:0]

- **Parameters**

uData [in] : Pre-sent data:0~0xFFFFFFFF

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/*Data transmission：MSB First, 8bit Send 0x55*/
DrvSPI32_Write(0x55<<24);
/*Data transmission：LSB First, 8bit Send 0x55*/
DrvSPI32_Write(0x55);
```

7.3.17. DrvSPI32_Enable

- **Prototype**

void DrvSPI32_Enable (void);

- **Description**

Enable the SPI function.

Configure the register 0x40F00[0]=1b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* Enable the SPI */
DrvSPI32_Enable();
```

7.3.18. DrvSPI32_BitLength

- **Prototype**

void DrvSPI32_BitLength (unsigned int uData);

- **Description**

Set the SPI transfer Bit length.

Configure the register 0x40F04[4:0]

- **Parameters**

uData[in] : Specify SPI data length. It could be 0x04~0x20

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  None

- **Example**

  /* Set SPI transfer Bit length 8*/

  DrvSPI32_BitLength(8);

7.3.19. DrvSPI32_GetDCFlag

- **Prototype**

  unsigned int DrvSPI32_GetDCFlag(void);

- **Description**

  Get data loss flag of state.

  Read the register 0x40F00[18]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

  0: Normal.

  1: Rx buffer data is overwritten.

- **Example**

  /* Check the status of DCF */

  unsigned char flag；

  flag=DrvSPI32_GetDCFlag();

7.3.20. DrvSPI32_IsABFlag

- **Prototype**

  unsigned int DrvSPI32_IsABFlag(void);

- **Description**

  Check whether the data deficient

  Read the register 0x40F00[20]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvSPI32.h

- **Return Vaule**

© 2021-2022HYCON Technology Corp

0: Normal.

1: SPI Bus receive data length is less than BL.

- **Example**

/* Check the status of ABF */

unsigned char flag；flag=DrvSPI32_IsABFlag();

7.3.21. DrvSPI32_IsOVFlag

- **Prototype**

unsigned int DrvSPI32_IsOVFlag(void);

- **Description**

Check whether the received data is too long

Read the register 0x40F00[21]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

0: Normal.

1: SPI Bus receive data length is greater than BL

- **Example**

/* Check the status of OVF */

unsigned char flag；flag=DrvSPI32_IsOVFlag();

7.3.22. DrvSPI32_IsRxFlag

- **Prototype**

unsigned int DrvSPI32_IsRxFlag(void);

- **Description**

Check whether the Rx buffer data in the update.

Read the register 0x40F00[22]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

0: Normal.

1: SPI buffer data update

- **Example**

© 2021-2022HYCON Technology Corp

```
/* Check the status of RxF */
unsigned char flag；flag=DrvSPI32_IsRxFlag();
```

7.3.23. DrvSPI32_SetEndian

- **Prototype**

```
void DrvSPI32_SetEndian(E_DRVSPI_ENDIAN eEndian);
```

- **Description**

Set the data transfer from the MSB or LSB start

Configure the register 0x40F04[18]

- **Parameters**

eEndian [in] : the input range is 0~1

1：Send LSB first

0：Send MSB first

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* The transfer order is LSB first */
DrvSPI32_SetEndian(E_DRVSPI_LSB_FIRST);
```

7.3.24. DrvSPI32_SetCSO

- **Prototype**

```
void DrvSPI32_SetCSO(E_DRVSPI_CS eCS);
```

- **Description**

Set the CS signal simulator control bit, Configure the register 0x40F04[20]

Note: The old function DrvSPI32_SetCS(E_DRVSPI_CS eCS) is the same operation as DrvSPI32_SetCSO(E_DRVSPI_CS eCS)

- **Parameters**

eCS[in]:

0: CS signal active low

1: CS signal active high

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* Set low level is effective */
DrvSPI32_SetCSO(E_DRVSPI_CSLow);
```

7.3.25. DrvSPI32_DisableIO

- **Prototype**

```
void DrvSPI32_DisableIO(void);
```

- **Description**

Disable the SPI port to transmit

Configure the register 0x40844[4]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvSPI32.h

- **Return Vaule**

None

- **Example**

```
/* Disable the SPI port to transmit */
DrvSPI32_DisableIO();
```

7.3.26. DrvSPI32_EnableIO

- **Prototype**

```
unsigned char DrvSPI32_EnableIO(uint32_t  uOuputPin);
```

- **Description**

Enable and specify the SPI port to transmit

Configure the register 0x40844[7:5] / 0x40844[4]=1;

- **Parameters**

uOuputPin [in]：specify the port as SPI, the effectively input range is 0~7.

0 : Port1.0 =CS, Port1.1 =CK, Port1.2 = DI, Port1.3 =DO

1 : Port1.4 =CS, Port1.5 =CK, Port1.6 = DI, Port1.7 =DO

2 : Port2.0 =CS, Port2.1 =CK, Port2.2 = DI, Port2.3 =DO

3 : Port2.4 =CS, Port2.5 =CK, Port2.6 = DI, Port2.7 =DO

4 : Port6.0 =CS, Port6.1 =CK, Port6.2 = DI, Port6.3 =DO

5 : Port7.4 =CS, Port7.5 =CK, Port7.6 = DI, Port7.7 =DO

6 : Port9.0 =CS, Port9.1 =CK, Port9.2 = DI, Port9.3 =DO

7 : Port8.0 =CS, Port8.1 =CK, Port8.2 = DI, Port8.3 =DO

- **Include**

Peripheral_lib/DrvSPI32.h

© 2021-2022HYCON Technology Corp

- **Return Vaule**

  None

- **Example**

  /* Enable the SPI port to transmit , select PT2.0~PT2.3*/

  DrvSPI32_EnableIO(2);

© 2021-2022HYCON Technology Corp

8. UART Driver

8.1. Introduction

The Universal Asynchronous Receiver/Transmitter (UART) performs a serial-to-parallel conversion on data characters received from the peripheral such as MODEM, and a parallel-to-serial conversion on data characters received from the CPU. Details please refer to the section in the target chip specification titled UART.

Item	Functions	Description
01	DrvUART_Open	Set UART1 module
02	DrvUART_Close	Close UART1 module
03	DrvUART_EnableInt	Enable the UART1 interrupt
04	DrvUART_GetTxFlag	Get the TX interrupt flag of UART1
05	DrvUART_GetRxFlag	Get the RX interrupt flag of UART1
06	DrvUART_ClrTxFlag	Clear the TX interrupt flag of UART1
07	DrvUART_ClrRxFlag	Clear the RX interrupt flag of UART1
08	DrvUART_Read	Read data from RCREG of UART1
09	DrvUART_ClrABDOVF	Clear the RXABDF flag of UART1
10	DrvUART_Write	Write data to TXREG of UART1
11	DrvUART_EnableWakeUp	Enable wake-up mode of UART1
12	DrvUART_DisableWakeUp	Disable wake-up mode of UART1
13	DrvUART_GetPERR	Get the PERR flag of UART1
14	DrvUART_GetFERR	Get the FERR flag of UART1
15	DrvUART_GetOERR	Get the OERR flag of UART1
16	DrvUART_GetABDOVF	Get the ABDOVF flag of UART1
17	DrvUART_Enable_AutoBaudrate	Enable Auto Baudrate of UART1
18	DrvUART_Disable_AutoBaudrate	Disable Auto Baudrate of UART1
19	DrvUART_CheckTRMT	Read the flag of Transmit Shift Register Status
20	DrvUART_ClkEnable	Enable and select the UART1 clock source
21	DrvUART_ClkDisable	Disable the UART1 clock source
22	DrvUART_Enable	Enable the UART1 function
23	DrvUART_ConfigIO	Enable and select the IO port as UART1 communication port
24	DrvUART_TRStatus	Read the RX/TX status of UART1
25	DrvUART_IntType	Set the interrupt trigger method of UART1 RX and TX
26	DrvUART_GetNERR	Get the RX Noise detected flag of UART1
27	DrvUART_ClrPERR	Clear the Parity Error flag of UART1
28	DrvUART_ClrFERR	Clear the RX Fram check error flag of UART1
29	DrvUART_ClrOERR	Clear the RX Buffer over run error flag of UART1
30	DrvUART_ClrNERR	Clear the RX Noise dected flag of UART1
31	DrvUART2_Open	Set UART2 module
32	DrvUART2_Enable	Enable the UART2 function
33	DrvUART2_Close	Disable the UART2 function
34	DrvUART2_EnableInt	Enable the UART2 TX or RX interrupt.
35	DrvUART2_IntType	Set the interrupt trigger method of

© 2021-2022HYCON Technology Corp

		UART2 RX and TX
36	DrvUART2_GetTxFlag	Get the Tx interrupt flag of UART2
37	DrvUART2_GetRxFlag	Get the Rx interrupt flag of UART2
38	DrvUART2_ClrTxFlag	Clear the Tx interrupt flag of UART2
39	DrvUART2_ClrRxFlag	Clear the Rx interrupt flag of UART2
40	DrvUART2_Read	Read data received from UART2
41	DrvUART2_Write	Write data to TXREG register of UART2
42	DrvUART2_EnableWakeUp	Enable wake-up mode of UART2
43	DrvUART2_DisableWakeUp	Disable wake-up mode of UART2
44	DrvUART2_Enable_AutoBaudrate	Enable Auto Baudrate of UART2
45	DrvUART2_Disable_AutoBaudrate	Disable Auto Baudrate of UART2
46	DrvUART2_GetPERR	Get the Parity Error flag of UART2
47	DrvUART2_GetFERR	Get the FERR flag of UART2
48	DrvUART2_GetOERR	Get the OERR flag of UART2
49	DrvUART2_GetNERR	Get the RX Noise detected flag of UART2
50	DrvUART2_ClrPERR	Clear the Parity Error flag of UART2
51	DrvUART2_ClrFERR	Clear the RX Fram check error flag of UART2
52	DrvUART2_ClrOERR	Clear the RX Buffer over run error flag of UART2
53	DrvUART2_ClrNERR	Clear the RX Noise dected flag of UART2
54	DrvUART2_GetABDOVF	Get the RXABDF flag of UART2
55	DrvUART2_ClrABDOVF	Clear the RXABDF flag
56	DrvUART2_TRStatus	Read the RX and TX status of UART2
57	DrvUART2_CheckTRMT	Read the UART2 flag of Transmit Shift Register Status (TXBF)
58	DrvUART2_ClkEnable	Enable and select the UART2 clock source
59	DrvUART2_ClkDisable	Disable the UART2 clock source
60	DrvUART2_ConfigIO	Enable and select the IO port as UART2 communication port

© 2021-2022HYCON Technology Corp

8.2. Type Definition

E_DATABITS_SETTINGS

Enumeration identifier	Value	Description
DRVUART_DATABITS_6	0x0	Word length select: Character length is 6 bits.
DRVUART_DATABITS_7	0x1	Word length select: Character length is 7 bits.
DRVUART_DATABITS_8	0x2	Word length select: Character length is 8 bits.
DRVUART_DATABITS_9	0x3	Word length select: Character length is 9 bits.

E_STOPBITS_SETTINGS

Enumeration identifier	Value	Description
DRVUART_STOPBITS_05	0x0	StopBits length selec:0.5 bits
DRVUART_STOPBITS _1	0x1	StopBits length selec:1 bits.
DRVUART_STOPBITS _15	0x2	StopBits length selec:1.5 bits
DRVUART_STOPBITS_2	0x3	StopBits length selec:2 bits.

E_PARITY_SETTINGS

Enumeration identifier	Value	Description
DRVUART_PARITY_NONE	0x0	None parity
DRVUART_PARITY_ODD	0x1	Odd parity enable
DRVUART_PARITY_EVEN	0x2	Even parity enable

E_BAUD_RATE_SETTINGS

Enumeration identifier	Value	Description
B1200	0x0	Baud rate=1200
B2400	0x1	Baud rate=2400
B4800	0x2	Baud rate=4800
B9600	0x3	Baud rate=9600
B14400	0x4	Baud rate=14400
B19200	0x5	Baud rate=19200
B38400	0x6	Baud rate=38400
B57600	0x7	Baud rate=57600
B115200	0x8	Baud rate=115200

E_UART_ERROR_MESSAGE

Enumeration identifier	Value	Description
E_UART_ERR_CLOCK	0x2	CLOCK Parameter input error
E_UART_ERR_BAUDRATE	0x3	Baud rate Parameter input error
E_UART_ERR_PARITY	0x4	Parity Parameter input error
E_UART_ERR_DATABIT	0x5	Data bit Parameter input error
E_UART_ERR_STOPBIT	0x6	StopBits length setting error
E_UART_ERR_OUTPIN	0x7	Output pin Parameter input error

© 2021-2022HYCON Technology Corp

8.3. Functions

8.3.1. DrvUART_Open

● Prototype

```
unsigned int DrvUART_Open (   unsigned int    uClock
                              E_RAUD_RATE_SETTINGS uBaudRate ,
                              E_PARITY_SETTINGS uParity,
                              E_DATABITS_SETTINGS uDataBits,
                              unsigned int    uStopBits,
                              unsigned int    uOuputPin );
```

● Description

Select the UART frequency value to used (Should be noted, oscillator clock soure HSXT or HSRC effects UART frequency value, UART divider also effects UART frequency value), to automatically calculate the value to register 0x40E08[15:0], according to input the required baud rate value,UART1 with bit set, set the UART data-bit, Stop-bit,set the UART output pin.

Configure the register 0x40E00[7:4], 0x40E00[2]=1, 0x40E00[0]=1, 0x40E04[1:0], 0x40E08[15:0], 0x40844[3:0].

● Parameters

uClock : Type UART frequency value in kHz Unit. The input value of UART is URCK frequency. URCK frequency is selected from external HSXT or internal HSRC clock source, and it goes throught UACD[3:0] divider. If UACD=1, URCK=HSXT(or HSRC). If UACD=2, URCK=HSXT/2(or HSRC/2) and so on.
The input range is 1000~20000

uBaudRate [in] : Type baud rate

uParity [in] : NONE/EVEN/ODD parity, It could be

0 : None parity

1 : Even parity

2 : Odd parity.

uDataBits[in] : data bit setting, It could be

0 : 6 data bits.

1 : 7 data bits.

2 : 8 data bits

3 : 9 data bits

uStopBits[in] : stop bit setting

0： 0.5 Bit 1: 1 Bit

2： 1.5 Bit 3： 2 Bit

uOuputPin [in] :

0 : Port 1.0 =TX, Port 1.1 =RX

1 : Port 1.4 =TX, Port 1.5 =RX

© 2021-2022HYCON Technology Corp

2 : Port 2.0 =TX, Port 2.1 =RX

3 : Port 2.4 =TX, Port 2.5 =RX

4 : Port 6.0 =TX, Port 6.1 =RX

5 : Port 7.4 =TX, Port 7.5 =RX

6 : Port 9.0 =TX, Port 9.1 =RX

7 : Port 8.0 =TX, Port 8.1 =RX

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

0: Success.

2 : Wrong clock setting

3 : Wrong baud rate setting

4: Wrong party setting

5: Wrong Data bit setting

6:Wrong Stop bit setting

7: Wrong output pin setting

- **Example**

```
/* Set UART baud rate115200bps, 8 data bits ,1 stop bit, and none parity. PT2.0/PT2.1 used as interface*/
  DrvUART_Open(4147,115200, DRVUART_PARITY_NONE ,DRVUART_DATABITS_8,1,2);
  Note : Because UART frequency value is 4.147MHz, so input value is 4147. The unit is kHz
```

8.3.2. DrvUART_Close

- **Prototype**

void DrvUART_Close (void);

- **Description**

Disable uart

Clear the register 0x40E00[2]=0, 0x40E00[0]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* Close UART */
DrvUART_Close();
```

8.3.3. DrvUART_EnableInt

- **Prototype**

  unsigned int DrvUART_EnableInt(unsigned int uTXIE, unsigned int uRXIE);

- **Description**

  Enable the UART1 TX or RX interrupt.

  Configure the register 0x40000[19:18]

- **Parameters**

  uTXIE [in] : UART1 Tx Interrupt

  0 : Disable

  1 : Enable

  uRXIE [in] : UART1 Rx Interrupt

  0 : Disable

  1 : Enable

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable the UART1 TX and RX interrupt */
DrvUART_EnableInt(1,1);
```

8.3.4. DrvUART_GetTxFlag

- **Prototype**

  unsigned int DrvUART_GetRxFlag (void);

- **Description**

  Get the Tx interrupt flag of UART1.

  Read the register 0x40000[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1: Interrupted

  0: Normal

- **Example**

```
/* Get the Tx interrupt flag. */
DrvUART_GetTxFlag();
```

© 2021-2022HYCON Technology Corp

8.3.5. DrvUART_GetRxFlag

- **Prototype**

  unsigned int DrvUART_GetRxFlag (void);

- **Description**

  Get the Rx interrupt flag of UART1.

  Read the register 0x40000[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1: Interrupted

  0: Normal

- **Example**

  /* Get the Rx interrupt flag. */

  unsigned char flag；flag=DrvUART_GetRxFlag();

8.3.6. DrvUART_ClrTxFlag

- **Prototype**

  void DrvUART_ClrTxFlag (void);

- **Description**

  Clear the Tx interrupt flag of UART1.

  Configure the register 0x40000[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

  /* Clear the Tx interrupt flag. */

  DrvUART_ClrTxFlag();

8.3.7. DrvUART_ClrRxFlag

- **Prototype**

  void DrvUART_ClrRxFlag (void);

© 2021-2022HYCON Technology Corp

- **Description**

  Clear the Rx interrupt flag of UART1.

  Configure the register 0x40000[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

  /* Clear the Rx interrupt flag. */

  DrvUART_ClrRxFlag();

8.3.8. DrvUART_Read

- **Prototype**

  unsigned int DrvUART_Read(void);

- **Description**

  Read data received from UART1.

  Read the register 0x40E0C[8:0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  The return value is RX Data buffer register data.

- **Example**

  /* Read the RX Data buffer register data. */

  unsined int rx_data；rx_data=DrvUART_Read();

8.3.9. DrvUART_ClrABDOVF

- **Prototype**

  unsigned int DrvUART_ClrABDOVF(void)

- **Description**

  Clear the RxABDF flag of UART1.

  Configure the register 0x40E04[4]

- **Parameters**

  None

- **Include**

© 2021-2022HYCON Technology Corp

Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* Clear the RxABDF flag */
DrvUART_ClrABDOVF();
```

8.3.10. DrvUART_Write

- **Prototype**

```
void DrvUART_Write(unsigned int uData);
```

- **Description**

  Write data to TX data register of UART1.

  Configure the register 0x40E0C[24:16]

- **Parameters**

  uData [in]

  data to be sent

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* Using UART1 to send one byte 0x55 */
DrvUART_Write(0x55);
```

8.3.11. DrvUART_EnableWakeUp

- **Prototype**

```
void DrvUART_EnableWakeUp(void);
```

- **Description**

  Enable wake-up mode of UART1

  Configure the register 0x40E04[2]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

© 2021-2022HYCON Technology Corp

```
/* Enable wake up. */
DrvUART_EnableWakeUp();
```

8.3.12. DrvUART_DisableWakeUp

- **Prototype**

```
void DrvUART_DisableWakeUp(void);
```

- **Description**

Disable wake-up mode of UART1

Configure the register 0x40E04[2]=0。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* Disable UART1 wake-up mode */
DrvUART_DisableWakeUp();
```

8.3.13. DrvUART_GetPERR

- **Prototype**

```
unsigned int DrvUART_GetPERR(void);
```

- **Description**

Get the Parity Error flag of UART1.

Read the register 0x40E00[20]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

1 : Parity error

0 : No parity error

- **Example**

```
/* Get the PERR flag. */
unsigned char flag; flag=DrvUART_GetPERR();
```

© 2021-2022HYCON Technology Corp

8.3.14. DrvUART_GetFERR

- **Prototype**

  unsigned int DrvUART_GetFERR(void);

- **Description**

  Get the FERR flag of UART1.

  Read the register 0x40E00[21]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1 : Framing error

  0 : No framing error

- **Example**

```
/* Get the FERR flag. */
unsigned char flag ; flag=DrvUART_GetFERR();
```

8.3.15. DrvUART_GetOERR

- **Prototype**

  unsigned int DrvUART_GetOERR(void);

- **Description**

  Get the OERR flag of UART1.

  Read the register 0x40E00[23]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1 : Overrun error

  0 : No overrun error

- **Example**

```
/* Get the OERR flag. */
unsigned char flag ; flag=DrvUART_GetOERR();
```

8.3.16. DrvUART_GetABDOVF

- **Prototype**

  unsigned int DrvUART_GetABDOVF(void);

© 2021-2022HYCON Technology Corp

- **Description**

  Get the RxABDF flag of UART1.

  Read the register 0x40E04[4]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1 : A BRG rollover has occurred during Auto-Baud Rate Detect mode

  0 : No BRG rollover has occurred

- **Example**

```
/* Get the RxABDF flag. */
unsigned char flag ; flag=DrvUART_GetABDOVF();
```

8.3.17. DrvUART_Enable_AutoBaudrate

- **Prototype**

  void DrvUART_Enable_AutoBaudrate (void);

- **Description**

  Enable Auto Baudrate of UART1

  Configure the register 0x40E04[3]=1

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* Enable Auto Baudrate */
DrvUART_Enable_AutoBaudrate();
```

8.3.18. DrvUART_Disable_AutoBaudrate

- **Prototype**

  void DrvUART_Disable_AutoBaudrate (void);

- **Description**

  Disable Auto Baudrate of UART1

  Configure the register 0x40E04[3]=0b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* Disable Auto Baudrate */

DrvUART_Disable_AutoBaudrate();

8.3.19. DrvUART_CheckTRMT

- **Prototype**

Unsigned int DrvUART_CheckTRMT

- **Description**

Read the UART1 flag of Transmit Shift Register Status(TXBF).

Read the register 0x40E00[18]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

0: Transmit Shift Register empty

1: Transmit Shift Register full

- **Example**

/* Read the flag of Transmit Shift Register Status */

DrvUART_Write(data) ;

While(DrvUART_CheckTRMT()) ;//wait until TXBF=0

8.3.20. DrvUART_ClkEnable

- **Prototype**

unsigned int DrvUART_ClkEnable(unsigned int uclk,unsigned int uprescale) ;

- **Description**

Enable and select the UART1 clock source . Specify the clock source divider

Configure the register 0x40308[21:16]

- **Parameters**

uclk[in] : EUART clock source

0 : External high speed oscillator

1 : Internal high speed oscillator

uprescale[in]：the clock source divider

© 2021-2022HYCON Technology Corp

0000 : EUART CLOCK SOURCE/1	1000 : Rsv
0001 : EUART CLOCK SOURCE/2	1001 : Rsv
0010 : EUART CLOCK SOURCE/4	1010 : Rsv
0011 : EUART CLOCK SOURCE/8	1011 : Rsv
0100 : EUART CLOCK SOURCE/16	1100 : Rsv
0101 : EUART CLOCK SOURCE/32	1101 : Rsv
0110 : EUART CLOCK SOURCE/64	1110 : Rsv
0111 : EUART CLOCK SOURCE/128	1111 : Rsv

● Include

Peripheral_lib/DrvUART.h

● Return Vaule

0: Operation successful

Other : Incorrect argument

● Example

```
/* select the external clock source, divider:clk/1 */
DrvUART_ClkEnable(0,0);
```

8.3.21. DrvUART_ClkDisable

● Prototype

Void DrvUART_ClkDisable(void) ;

● Description

Disable the UART1 clock source.

Configure the register 0x40308[20]=0

● Parameters

None

● Include

Peripheral_lib/DrvUART.h

● Return Vaule

None

● Example

```
/* Disable the UART1 clock source */
DrvUART_ClkDisable();
```

8.3.22. DrvUART_Enable

● Prototype

Void DrvUART_Enable(void) ;

● Description

© 2021-2022HYCON Technology Corp

Enable the UART1 function

Configure the register 0x40E00[2]=1, 0x40E00[0]=1

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* Enable the UART1 clock source */
DrvUART_Enable();
```

8.3.23. DrvUART_ConfigIO

- **Prototype**

unsigned char DrvUART_ConfigIO(unsigned char ioen,unsigned int uOuputPin) ;

- **Description**

Enable and select the IO port as UART1 communication port

Configure the register 0x40844[3:0]

- **Parameters**

ioen[in] : EURAT1 input/output to port enable control

0 : disable

1 : enable

uoutputPin[in] : select the UART1 communication port

0 : Port 1.0 =TX, Port 1.1 =RX

1 : Port 1.4 =TX, Port 1.5 =RX

2 : Port 2.0 =TX, Port 2.1 =RX

3 : Port 2.4 =TX, Port 2.5 =RX

4 : Port 6.0 =TX, Port 6.1 =RX

5 : Port 7.4 =TX, Port 7.5 =RX

6 : Port 9.0 =TX, Port 9.1 =RX

7 : Port 8.0 =TX, Port 8.1 =RX

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* enable and select PT2.0/PT2.1 as UART port*/
DrvUART_ConfigIO(1,2);
```

© 2021-2022HYCON Technology Corp

8.3.24. DrvUART_TRStatus

- **Prototype**

  Unsigned int DrvUART_TRStatus(unsigned int uMode)

- **Description**

  Read the RX and TX status of UART1, read the register 0x40E00[19:16]

- **Parameters**

  uMode[in] :

  0 : RXBF; 1 : RXBUSY; 2 : TXBF; 3 : TXBUSY

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: idle; 1: Busy (for TXBUSY and RXBUSY)

  0: empty; 1: full (for TXBF and RXBF)

- **Example**

```
/* Get the TXBF flag. */
DrvUART_Write(data) ;
While(DrvUART_TRStatus(2)) ;//wait until TXBF=0
```

8.3.25. DrvUART_IntType

- **Prototype**

  unsigned int DrvUART_IntType(unsigned int uTXIT, unsigned int uRXIT)

- **Description**

  Set the interrupt trigger method of UART1 RX and TX .

  Configure the register 0x40E00[1]/ 0x40E00[3]

- **Parameters**

  uTXIT [in] : the interrupt trigger method of TX

  0 : Send out the interrupt when the Tx Data Buffer is idle, and the interrupt disappears after the data are written in.

  1 : Sent out the interrupt after one piece of data is transmitted by the Tx

  uRXIT[in] : the interrupt trigger method of RX

  0 : Send out the interrupt when the Rx Data Buffer has data, and the interrupt disappears after the data are read.

  1 : Send out the interrupt after one piece of data is received by the Rx.

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: Operation successful

© 2021-2022HYCON Technology Corp

Other : Incorrect argument

- **Example**

/* Send out the interrupt when the Tx Data Buffer is idle and the Rx Data Buffer has data */

DrvUART_IntType(0, 0) ;

8.3.26. DrvUART_GetNERR

- **Prototype**

unsigned int DrvUART_GetNERR(void);

- **Description**

Get the RX Noise detected flag of UART1.

Read the register 0x40E00[22]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

1 : Noise detected

0 : Normal

- **Example**

/* Get the NERR flag. */

unsigned char flag; flag=DrvUART_GetNERR();

8.3.27. DrvUART_ClrPERR

- **Prototype**

void DrvUART_ClrPERR(void);

- **Description**

Clear the Parity Error flag of UART1, configure the register 0x40E00[20]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* clear the PERR flag. */

DrvUART_ClrPERR();

© 2021-2022HYCON Technology Corp

8.3.28. DrvUART_ClrFERR

- **Prototype**

  void DrvUART_ClrFERR(void);

- **Description**

  Clear the RX Fram check error flag of UART1.

  Configure the register 0x40E00[21]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* clear the FERR flag. */
DrvUART_ClrFERR();
```

8.3.29. DrvUART_ClrOERR

- **Prototype**

  void DrvUART_ClrOERR(void);

- **Description**

  Clear the RX Buffer over run error flag of UART1.

  Configure the register 0x40E00[23]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* clear the OERR flag. */
DrvUART_ClrOERR();
```

8.3.30. DrvUART_ClrNERR

- **Prototype**

  void DrvUART_ClrNERR(void);

- **Description**

  Clear the RX Noise dected flag of UART1.

© 2021-2022HYCON Technology Corp

Configure the register 0x40E00[22]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* clear the NERR flag. */
DrvUART_ClrNERR();
```

8.3.31. DrvUART2_Open

- **Prototype**

```
unsigned int DrvUART2_Open (    unsigned int    uClock
                                E_RAUD_RATE_SETTINGS    uBaudRate ,
                                E_PARITY_SETTINGS       uParity,
                                E_DATABITS_SETTINGS     uDataBits,
                                unsigned int            uStopBits,
                                unsigned int            uOuputPin    );
```

- **Description**

Select the UART2 frequency value to used (Should be noted, oscillator clock soure HSXT or HSRC effects UART2 frequency value, UART2 divider also effects UART2 frequency value), to automatically calculate the value to 0x40E18[15:0] , according to input the required baud rate value,UART2 with bit set, set the UART2 data-bit, Stop-bit,set the UART output pin

Configure the register 0x40E10[7:4], 0x40E10[2]=1, 0x40E10[0]=1, 0x40E14[1:0];
0x40E18[15:0]; 0x4084C[3:0].

- **Parameters**

uClock : Type UART2 frequency value in kHz Unit. The input value of UART2 is UR2CK frequency. UR2CK frequency is selected from external HSXT or internal HSRC clock source, and it goes throught UA2CD[3:0] divider. If UA2CD=1, UR2CK=HSXT(or HSRC). If UA2CD=2, UR2CK=HSXT/2(or HSRC/2) and so on.
The input range is 1000~20000

uBaudRate [in] : Type baud rate

uParity [in] : NONE/EVEN/ODD parity. It could be

0 : None parity

1 : Even parity

2 : Odd parity.

uDataBits[in] : data bit setting. It could be

0 : 6 data bits.

© 2021-2022HYCON Technology Corp

1 : 7 data bits.

2 : 8 data bits

3 : 9 data bits

uStopBits[in] : stop bit setting

0： 0.5 Bit 1: 1 Bit

2： 1.5 Bit 3： 2 Bit

uOuputPin [in] :

0 : Port 1.2 =TX2, Port 1.3 =RX2

1 : Port 1.6 =TX2, Port 1.7 =RX2

2 : Port 2.2 =TX2, Port 2.3 =RX2

3 : Port 2.6 =TX2, Port 2.7 =RX2

4 : Port 6.2 =TX2, Port 6.3 =RX2

5 : Port 7.6 =TX2, Port 7.7 =RX2

6 : Port 9.2 =TX2, Port 9.3 =RX2

7 : Port 8.2 =TX2, Port 8.3 =RX2

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

0: Success.

2 : Wrong clock setting

3 : Wrong baud rate setting

4: Wrong party setting

5: Wrong Data bit setting

6:Wrong Stop bit setting

7: Wrong output pin setting

- **Example**

/* Set UART2 baud rate 115200bps, 8 data bits ,1 stop bit, and none parity. PT2.2/PT2.3 used as interface*/

DrvUART2_Open(4147,115200, DRVUART_PARITY_NONE ,DRVUART_DATABITS_8,1,2);

Note : Because UART2 frequency value is 4.147MHz, so input value is 4147. The unit is kHz

8.3.32. DrvUART2_Enable

- **Prototype**

Void DrvUART2_Enable(void) ;

- **Description**

Enable the UART2 function

Configure the register 0x40E10[2]=1 · 0x40E10[0]=1

- **Parameters**

© 2021-2022HYCON Technology Corp

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* Enable the UART2 */

DrvUART2_Enable();

8.3.33. DrvUART2_Close

- **Prototype**

void DrvUART2_Close (void);

- **Description**

Disable UART2, clear the register 0x40E10[2]=0, 0x40E10[0]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* Close UART2 */

DrvUART2_Close();

8.3.34. DrvUART2_EnableInt

- **Prototype**

unsigned int DrvUART2_EnableInt(unsigned int uTXIE, unsigned int uRXIE);

- **Description**

Enable the UART2 TX or RX interrupt.

Configure the register 0x40018[19:18]

- **Parameters**

uTXIE [in] : UART2 Tx Interrupt

0 : Disable

1 : Enable

uRXIE [in] : UART2 Rx Interrupt

0 : Disable

1 : Enable

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable the UART2 TX and RX interrupt */
DrvUART2_IntType(0,0);    //set the interrupt trigger method of UART2
DrvUART2_EnableInt(1,1);
```

8.3.35. DrvUART2_IntType

- **Prototype**

  Unsigned int DrvUART2_IntType(unsigned int uTXIT, unsigned int uRXIT)

- **Description**

  Set the interrupt trigger method of UART2 RX and TX .

  Configure the register 0x40E10[1]/ 0x40E10[3]

- **Parameters**

  uTXIT [in] : the interrupt trigger method of TX

  0 : Send out the interrupt when the Tx Data Buffer is idle, and the interrupt disappears after the data are written in.

  1 : Sent out the interrupt after one piece of data is transmitted by the Tx

  uRXIT[in] : the interrupt trigger method of RX

  0 : Send out the interrupt when the Rx Data Buffer has data, and the interrupt disappears after the data are read.

  1 : Send out the interrupt after one piece of data is received by the Rx.

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: Operation successful

  1: Incorrect argument

- **Example**

```
/* Send out the interrupt when the Tx Data Buffer is idle and the Rx Data Buffer has data   */
DrvUART2_IntType(0, 0) ;
```

8.3.36. DrvUART2_GetTxFlag

- **Prototype**

  unsigned int DrvUART2_GetTxFlag (void);

- **Description**

  Get the Tx interrupt flag of UART2.

  Read the register 0x40018[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1: Interrupted

  0: Normal

- **Example**

```
/* Get the Tx interrupt flag. */
DrvUART2_GetTxFlag();
```

8.3.37. DrvUART2_GetRxFlag

- **Prototype**

  unsigned int DrvUART2_GetRxFlag (void);

- **Description**

  Get the Rx interrupt flag of UART2.

  Read the register 0x40018[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1: Interrupted

  0: Normal

- **Example**

```
/* Get the Rx interrupt flag. */
unsigned char flag；flag=DrvUART2_GetRxFlag();
```

8.3.38. DrvUART2_ClrTxFlag

- **Prototype**

  void DrvUART2_ClrTxFlag (void);

- **Description**

  Clear the Tx interrupt flag of UART2.

  Configure the register 0x40018[3]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

  /* Clear the Tx interrupt flag. */

  DrvUART2_ClrTxFlag();

8.3.39. DrvUART2_ClrRxFlag

- **Prototype**

  void DrvUART2_ClrRxFlag (void);

- **Description**

  Clear the Rx interrupt flag of UART2.

  Configure the register 0x40018[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

  /* Clear the Rx interrupt flag. */

  DrvUART2_ClrRxFlag();

8.3.40. DrvUART2_Read

- **Prototype**

  unsigned int DrvUART2_Read(void);

- **Description**

Read data received from UART2.

Read the register 0x40E1C[8:0]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

The return value is RCREG register data.

- **Example**

/* Read the RCREG register data. */

unsined int rx_data；rx_data=DrvUART2_Read();

8.3.41. DrvUART2_Write

- **Prototype**

void DrvUART2_Write(unsigned int uData);

- **Description**

Write data to TXREG register of UART2.

Configure the register 0x40E1C[24:16]

- **Parameters**

uData [in] : data to be sent

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* using UART2 to send one byte 0x55 */

DrvUART2_Write(0x55);

8.3.42. DrvUART2_EnableWakeUp

- **Prototype**

void DrvUART2_EnableWakeUp(void);

- **Description**

Enable wake-up mode of UART2

Configure the register 0x40E14[2]=1

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* Enable wake up. */

DrvUART2_EnableWakeUp();

© 2021-2022HYCON Technology Corp

8.3.43. DrvUART2_DisableWakeUp

- **Prototype**

  void DrvUART2_DisableWakeUp(void);

- **Description**

  Disable wake-up mode of UART2

  Configure the register 0x40E14[2]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* Disable wake up. */
DrvUART2_DisableWakeUp();
```

8.3.44. DrvUART2_Enable_AutoBaudrate

- **Prototype**

  void DrvUART2_Enable_AutoBaudrate (void);

- **Description**

  Enable Auto Baudrate of UART2

  Configure the register 0x40E14[3]=1

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

```
/* Enable Auto Baudrate */
DrvUART2_Enable_AutoBaudrate();
```

8.3.45. DrvUART2_Disable_AutoBaudrate

- **Prototype**

  void DrvUART2_Disable_AutoBaudrate (void);

- **Description**

© 2021-2022HYCON Technology Corp

Disable Auto Baudrate of UART2

Configure the register 0x40E14[3]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* Enable Auto Baudrate */
DrvUART2_Disable_AutoBaudrate();
```

8.3.46. DrvUART2_GetPERR

- **Prototype**

unsigned int DrvUART2_GetPERR(void);

- **Description**

Get the Parity Error flag of UART2.

Read the register 0x40E10[20]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

1 : Parity error

0 : No parity error

- **Example**

```
/* Get the PERR flag. */
unsigned char flag; flag=DrvUART2_GetPERR();
```

8.3.47. DrvUART2_GetFERR

- **Prototype**

unsigned int DrvUART2_GetFERR(void);

- **Description**

Get the FERR flag of UART2.

Configure the register 0x40E10[21]

- **Parameters**

None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1 : Framing error

  0 : No framing error

- **Example**

```
/* Get the FERR flag. */
unsigned char flag ; flag=DrvUART2_GetFERR();
```

8.3.48. DrvUART2_GetOERR

- **Prototype**

```
unsigned int DrvUART2_GetOERR(void);
```

- **Description**

  Get the OERR flag of UART2.

  Configure the register 0x40E10[23]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  1 : Overrun error

  0 : No overrun error

- **Example**

```
/* Get the OERR flag. */
unsigned char flag ; flag=DrvUART2_GetOERR();
```

8.3.49. DrvUART2_GetNERR

- **Prototype**

```
unsigned int DrvUART2_GetNERR(void);
```

- **Description**

  Get the RX Noise detected flag of UART2.

  Read the register 0x40E10[22]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

© 2021-2022HYCON Technology Corp

1 : Noise detected

0 : Normal

- **Example**

/* Get the NERR flag. */

unsigned char flag; flag=DrvUART2_GetNERR();

8.3.50. DrvUART2_ClrPERR

- **Prototype**

void DrvUART2_ClrPERR(void);

- **Description**

Clear the Parity Error flag of UART2.

Configure the register 0x40E10[20]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* clear the PERR flag. */

DrvUART2_ClrPERR();

8.3.51. DrvUART2_ClrFERR

- **Prototype**

void DrvUART2_ClrFERR(void);

- **Description**

Clear the RX Fram check error flag of UART2.

Configure the register 0x40E10[21]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

/* clear the FERR flag. */

DrvUART2_ClrFERR();

© 2021-2022HYCON Technology Corp

8.3.52. DrvUART2_ClrOERR

- **Prototype**

  void DrvUART2_ClrOERR(void);

- **Description**

  Clear the RX Buffer over run error flag of UART2.

  Configure the register 0x40E10[23]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

  /* clear the OERR flag. */

  DrvUART2_ClrOERR();

8.3.53. DrvUART2_ClrNERR

- **Prototype**

  void DrvUART2_ClrNERR(void);

- **Description**

  Clear the RX Noise dected flag of UART2.

  Configure the register 0x40E10[22]=0

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  None

- **Example**

  /* clear the NERR flag. */

  DrvUART2_ClrNERR();

8.3.54. DrvUART2_GetABDOVF

- **Prototype**

  unsigned int DrvUART2_GetABDOVF(void);

- **Description**

  Get the RxABDF flag of UART2.

© 2021-2022HYCON Technology Corp

Configure the register 0x40E14[4]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

1 : A BRG rollover has occurred during Auto-Baud Rate Detect mode

0 : No BRG rollover has occurred

- **Example**

```
/* Get the RxABDF flag. */
unsigned char flag ; flag=DrvUART2_GetABDOVF();
```

8.3.55. DrvUART2_ClrABDOVF

- **Prototype**

unsigned int DrvUART2_ClrABDOVF(void)

- **Description**

Clear the RXABDF flag.

Configure the register 0x40E14[4]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* Clear the RXABDF flag */
DrvUART2_ClrABDOVF();
```

8.3.56. DrvUART2_TRStatus

- **Prototype**

Unsigned int DrvUART2_TRStatus(unsigned int uMode)

- **Description**

Read the RX and TX status of UART2.

Read the register 0x40E10[19:16]

- **Parameters**

uMode[in] :

0 : RXBF; 1 : RXBUSY; 2 : TXBF; 3 : TXBUSY

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  TXBUSY : 0 idle; 1 Busy

  TXBF : 0 empty; 1 full

  RXBUSY: 0 idle; 1 Busy

  RXBF : 0 empty; 1 full

- **Example**

```
/* Get the TXBF flag. */
DrvUART2_Write(data) ;
While(DrvUART2_TRStatus(2)) ;//wait until TXBF=0
```

8.3.57. DrvUART2_CheckTRMT

- **Prototype**

  Unsigned int DrvUART2_CheckTRMT

- **Description**

  Read the UART2 flag of Transmit Shift Register Status(TXBF).

  Read the register 0x40E10[18]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: Transmit Shift Register empty

  1: Transmit Shift Register full

- **Example**

```
/* Read the flag of Transmit Shift Register Status */
DrvUART2_Write(data) ;
While(DrvUART2_CheckTRMT()) ;//wait until TXBF=0
```

8.3.58. DrvUART2_ClkEnable

- **Prototype**

  unsigned int DrvUART2_ClkEnable(unsigned int uclk,unsigned int uprescale) ;

- **Description**

  Enable and select the UART2 clock source . Specify the clock source divider

  Configure the register 0x40310[21:20]/ 0x40310[18 :16]

- **Parameters**

uclk[in] : EUART2 clock source

0 : External high speed oscillator

1 : Internal high speed oscillator

uprescale[in] : the clock source divider

0000 : EUART2 CLOCK SOURCE/1

0001 : EUART2 CLOCK SOURCE/2

0010 : EUART2 CLOCK SOURCE/4

0011 : EUART2 CLOCK SOURCE/8

0100 : EUART2 CLOCK SOURCE/16

0101 : EUART2 CLOCK SOURCE/32

0110 : EUART2 CLOCK SOURCE/64

0111 : EUART2 CLOCK SOURCE/128

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* select the external clock source, divider:clk/1 */
DrvUART2_ClkEnable(0,0);
```

8.3.59. DrvUART2_ClkDisable

- **Prototype**

Void DrvUART2_ClkDisable(void) ;

- **Description**

Disable the UART2 clock source.

Configure the register 0X40310[20]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvUART.h

- **Return Vaule**

None

- **Example**

```
/* Disable the UART2 clock source */
DrvUART2_ClkDisable();
```

8.3.60. DrvUART2_ConfigIO

- **Prototype**

  unsigned char DrvUART2_ConfigIO(unsigned char ioen,unsigned int uOuputPin) ;

- **Description**

  Enable and select the IO port as UART2 communication port

  Configure the register 0x4084C[3:0]

- **Parameters**

  ioen[in] : EURAT1 input/output to port enable control

  0 : disable

  1 : enable

  uoutputPin[in] : select the UART2 communication port

  0 : Port 1.2 =TX2, Port 1.3 =RX2

  1 : Port 1.6 =TX2, Port 1.7 =RX2

  2 : Port 2.2 =TX2, Port 2.3 =RX2

  3 : Port 2.6 =TX2, Port 2.7 =RX2

  4 : Port 6.2 =TX2, Port 6.3 =RX2

  5 : Port 7.6 =TX2, Port 7.7 =RX2

  6 : Port 9.2 =TX2, Port 9.3 =RX2

  7 : Port 8.2 =TX2, Port 8.3 =RX2

- **Include**

  Peripheral_lib/DrvUART.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* enable and select PT2.2/PT2.3 as UART2 port*/
DrvUART2_ConfigIO(1,2);
```

© 2021-2022HYCON Technology Corp

9. PMU Driver

9.1. Introduction

The following functions are included in Power Manager Section.

Item	Functions	Description
01	DrvPMU_VDD15Trim	Trim VDD15 voltage
02	DrvPMU_VDDA_Voltage	VDDA voltage selection
03	DrvPMU_VDDA_LDO_Ctrl	VDDA LDO enable control
04	DrvPMU_BandgapEnable	Band gap enable control
05	DrvPMU_BandgapDisable	Band gap disable control
06	DrvPMU_REFO_Enable	Reference buffer enable
07	DrvPMU_REFO_Disable	Reference buffer disable
08	DrvPMU_AnalogGround	ADC analog ground source
09	DrvPMU_LDO_LowPower	VDD LDO low power
10	DrvPMU_GetLVDO	Get LVD status
11	DrvPMU_EnableENLVD	Enable LVD function
12	DrvPMU_DisableENLVD	Disable LVD function
13	DrvPMU_SetLVDS	LVD detect value
14	DrvPMU_SetLVDVS	LVD comparator positive input
15	DrvPMU_SetLVD12	LVD comparator negative input
16	DrvPMU_LVDIntTriMode	LVD interrupt mode
17	DrvPMU_EnableLVDInt	Enable LVD interrupt
18	DrvPMU_DisableLVDInt	Disable LVD interrupt
19	DrvPMU_ClrLVDIF	Clear the interrupt flag of LVD
20	DrvPMU_EnableBOR2	Enable BOR2 function
21	DrvPMU_DisableBOR2	Disable BOR2 function
22	DrvPMU_BOR2_Mode	BOR2 operating mode
23	DrvPMU_DetectVol	BOR2 detect value
24	DrvPMU_ReadBOR2Status	Read BOR2 status

9.2. Type Definition

E_VDDA_OUTPUT_VOLTAGE

Enumeration identifier	Value	Description
E_VDDA2_4	0x0	Select the VDDA voltage of 2.4V
E_VDDA2_6	0x1	Select the VDDA voltage of 2.6V
E_VDDA2_9	0x2	Select the VDDA voltage of 2.9V
E_VDDA3_2	0x3	Select the VDDA voltage of 3.2V

E_VDDA_LDO_ENABLE_CONTROL

Enumeration identifier	Value	Description
E_HighZ	0x0	Select the VDDA voltage of 0V
E_LDO	0x1	Select the VDDA voltage of 2.4~3.3V

E_LVD_LVDS

Enumeration identifier	Value	Description
LVD_OFF	0	LVD Off
LVD_20V	1	Select the LVD voltage of 2.0V
LVD_21V	2	Select the LVD voltage of 2.1V

© 2021-2022HYCON Technology Corp

LVD_22V	3	Select the LVD voltage of 2.2V
LVD_23V	4	Select the LVD voltage of 2.3V
LVD_24V	5	Select the LVD voltage of 2.4V
LVD_25V	6	Select the LVD voltage of 2.5V
LVD_26V	7	Select the LVD voltage of 2.6V
LVD_27V	8	Select the LVD voltage of 2.7V
LVD_28V	9	Select the LVD voltage of 2.8V
LVD_29V	10	Select the LVD voltage of 2.9V
LVD_30V	11	Select the LVD voltage of 3.0V
LVD_33V	12	Select the LVD voltage of 3.3V
LVD_36V	13	Select the LVD voltage of 3.6V
LVD_40V	14	Select the LVD voltage of 4.0V
LVD_LVDIN	15	Select the LVD voltage of LVDIN

E_BOR2_BOR2TH

Enumeration identifier	Value	Description
BOR2_17V	0	Select the BOR2TH of 1.7v
BOR2_20V	1	Select the BOR2TH of 2.0v
BOR2_22V	2	Select the BOR2TH of 2.2v
BOR2_25V	3	Select the BOR2TH of 2.5v
BOR2_27V	4	Select the BOR2TH of 2.7v
BOR2_30V	5	Select the BOR2TH of 3.0v
BOR2_36V	6	Select the BOR2TH of 3.6v
BOR2_40V	7	Select the BOR2TH of 4.0v

© 2021-2022HYCON Technology Corp

9.3. Functions

9.3.1. DrvPMU_VDD15Trim

- **Prototype**

  unsigned int DrvPMU_VDD15Trim (void)

- **Description**

  Recommended to execute VDD15 Trim function first, and then change the operating frequency of the chip to ensure that the digital circuit is normal.

  Configure the register 0x40400[23:20].

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/* Execute VDD15 Trim function. */
DrvPMU_VDD15Trim();
```

9.3.2. DrvPMU_VDDA_Voltage

- **Prototype**

  unsigned int DrvPMU_VDDA_Voltage(E_VDDA_OUTPUT_VOLTAGE uVoltage)

- **Description**

  VDDA output voltage selection

  Configure the register 0x40400[19:18]

- **Parameters**

  uVoltage [in] : VDDA voltage selection, the input range is 0~3

  0 : 2.4V

  1 : 2.7V

  2 : 3.0V

  3 : 3.3V

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Select the VDDA voltage of 2.7V. */
```

© 2021-2022HYCON Technology Corp

```
DrvPMU_VDDA_Voltage(E_VDDA2_7);
```

9.3.3. DrvPMU_VDDA_LDO_Ctrl

- **Prototype**

  unsigned int DrvPMU_VDDA_LDO_Ctrl(E_VDDA_LDO_ENABLE_CONTROL uCtrl)

- **Description**

  VDDA LDO enable control.

  Configure the register 0x40400[17:16]

- **Parameters**

  uCtrl [in] :

  0 : High Z, VDDA=0

  3 : LDO, VDDA output voltage regulation

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Select the VDDA LDO enable. VDDA=2.7V*/
DrvPMU_VDDA_LDO_Ctrl(E_LDO);
DrvPMU_VDDA_Voltage(E_VDDA2_7);
```

9.3.4. DrvPMU_BandgapEnable

- **Prototype**

  void DrvPMU_BandgapEnable(void)

- **Description**

  Bandgap enable control.

  Configure the register 0x40400[4]=1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/* Enable bandgap. */
DrvPMU_BandgapEnable();
```

© 2021-2022HYCON Technology Corp

9.3.5. DrvPMU_BandgapDisable

- **Prototype**

  void DrvPMU_BandgapDisable(void)

- **Description**

  Bandgap disable control.

  Configure the register 0x40400[4]=0b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/* Disable bandgap. */
DrvPMU_BandgapDisable();
```

9.3.6. DrvPMU_REFO_Enable

- **Prototype**

  void DrvPMU_REFO_Enable(void)

- **Description**

  Reference buffer enable control.

  The output voltage is 1.2V. Need to enable the Bandgap.

  Configure the register 0x40400[1] =1b

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/* Enable REFO. */
DrvPMU_BandgapEnable() ; // enable the Bandgap
DrvPMU_REFO_Enable(); //Enable REFO
```

9.3.7. DrvPMU_REFO_Disable

- **Prototype**

  void DrvPMU_REFO_Disable(void)

© 2021-2022HYCON Technology Corp

- **Description**

  Reference buffer disable control.

  Configure the register 0x40400[1] =0b

- **Parameters**

  one

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/* Disable REFO. */
DrvPMU_REFO_Disable();
```

9.3.8. DrvPMU_AnalogGround

- **Prototype**

  unsigned int DrvPMU_AnalogGround(uAG)

- **Description**

  ADC analog ground source selection.

  Configure the register 0x40400[3]

- **Parameters**

  uAG [in]

  0 : External

  1 : Enable buffer and use internal source(need to work with ADC)

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Select the analog ground of external. */
DrvPMU_AnalogGround(0);
```

9.3.9. DrvPMU_LDO_LowPower

- **Prototype**

  unsigned int DrvPMU_LDO_LowPower(uLP)

- **Description**

  VDD LDO with low power control.

Configure the register 0x40400[0]

- **Parameters**

uLP [in]

0 : Normal(form sleep mode make up needs to set 0)

1 : Low power

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable the LDO of low power. */
DrvPMU_LDO_LowPower(1);
```

9.3.10. DrvPMU_GetLVDO

- **Prototype**

unsigned int DrvPMU_GetLVDO (void)

- **Description**

Get LVD status.

Read the register 0x40408[16]

- **Parameters**

None

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0： negative>positive

1: positive>negative

- **Example**

```
/*Read LVD status */
unsigned char Status;
Status= DrvPMU_GetLVDO ();
```

9.3.11. DrvPMU_EnableENLVD

- **Prototype**

unsigned int DrvPMU_EnableENLVD (void)

- **Description**

Enable LVD function

Configure the register 0x40408[0]=1.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

  /* Enable LVD function */

  DrvPMU_ EnableENLVD ();

9.3.12. DrvPMU_DisableENLVD

- **Prototype**

  unsigned int DrvPMU_DisableENLVD (void)

- **Description**

  Disable LVD function

  Configure the register 0x40408[0]=0.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

  /*Disable LVD function */

  DrvPMU_ DisableENLVD ();

9.3.13. DrvPMU_SetLVDS

- **Prototype**

  unsigned int DrvPMU_SetLVDS (uVoltage)

- **Description**

  Select the LVD detect voltage

  Configure the register 0x40408[7:4].

- **Parameters**

  uVoltage [in] :

  0 : OFF 1 : 2.0V

  2 : 2.1V 3 : 2.2V

  4 : 2.3V 5 : 2.4V

© 2021-2022HYCON Technology Corp

6 : 2.5V 7 : 2.6V

8 : 2.7V 9 : 2.8V

10 : 2.9V 11 : 3.0 V

12 : 3.3V 13 : 3.6V

14 : 4.0V 15 : LVDIN

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/*Set LVD voltage 2.1V */
DrvPMU_ SetLVDS (LVD_21V) ;
```

9.3.14. DrvPMU_SetLVDVS

- **Prototype**

unsigned int DrvPMU_ SetLVDVS (uLVDVS)

- **Description**

Select LVD positive input source

Configure the register 0x40408[1].

- **Parameters**

uVoltage [in] :

0 : VDD5V

1 : VLCD

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the LVD positive input VDD5V */
DrvPMU_ SetLVDVS (LVD_VDD5V) ;
```

9.3.15. DrvPMU_SetLVD12

- **Prototype**

unsigned int DrvPMU_ SetLVD12 (uLVD12)

- **Description**

Select LVD negative input source

Configure the register 0x40408[2].

- **Parameters**

uVoltage [in] :

0 : V12_BOR

1 : V12_BGR

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Set the LVD negative input BGR 1.2V */
DrvPMU_ SetLVD12 (LVD_V12_BGR) ;
```

9.3.16. DrvPMU_LVDIntTriMode

- **Prototype**

unsigned int DrvPMU_ LVDIntTriMode (uLVDITT)

- **Description**

LVD Interrupt Trigger Settings.

Configure the register 0x40408[3].

- **Parameters**

uLVDITT [in] :LVD Trigger Settings.

0 : LVDO 1->0

1 : LVDO 0->1

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* LVDO signal 1 to 0, trigger LVD interrupt */
DrvPMU_ LVDIntTriMode (0) ;
```

9.3.17. DrvPMU_EnableLVDInt

- **Prototype**

void DrvPMU_EnableLVDInt (void)

- **Description**

© 2021-2022HYCON Technology Corp

Enable LVD Interrupt function

Configure the register 0x4000C [16]=1.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/*Enable LVD Interrupt */
DrvPMU_EnableLVDInt();
```

9.3.18. DrvPMU_DisableLVDInt

- **Prototype**

  void DrvPMU_DisableLVDInt (void)

- **Description**

  Disable LVD Interrupt function

  Configure the register 0x4000C [16]=0.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  None

- **Example**

```
/*Disable LVD Interrupt */
DrvPMU_DisableLVDInt();
```

9.3.19. DrvPMU_ClrLVDIF

- **Prototype**

  void DrvPMU_ ClrLVDIF (void)

- **Description**

  Clear LVDIF Interrupt flag

  Configure the register 0x40408[3]=0.

- **Parameters**

  None

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/*Clear LVDIF Interrupt flag*/
DrvPMU_ ClrLVDIF (0) ;
```

9.3.20. DrvPMU_EnableBOR2

- **Prototype**

void DrvPMU_EnableBOR2(unsigned char utime)

- **Description**

Enable BOR2 function.

Configure the register 0x4040C[0]=1.

- **Parameters**

utime [in] :set BOR2 delay time. Range : 0~255, Each add 1, the delay time will increase by about 150us. It is recommended to enter the value "1".

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

None

- **Example**

```
/*Enable BOR2 */
DrvPMU_ EnableBOR2 (1);     //delay time must more than150us
```

9.3.21. DrvPMU_DisableBOR2

- **Prototype**

void DrvPMU_DisableBOR2 (void)

- **Description**

Disable BOR2 function

Configure the register 0x4040C[0]=0.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

None

© 2021-2022HYCON Technology Corp

- **Example**

```
/*Disable BOR2功能 */
DrvPMU_ DisableBOR2 ();
```

9.3.22. DrvPMU_BOR2_Mode

- **Prototype**

unsigned int DrvPMU_BOR2_Mode (uBOR2S)

- **Description**

Set BOR2 operating mode

Configure the register 0x4040C[1]

- **Parameters**

uBOR2S [in] :

0 : Interrupt mode

1 : Reset mode, Reset to 1 when BOR1 occurs.

- **Include**

Peripheral_lib/DrvPMU.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/*set BOR2 Reset mode */
DrvPMU_ BOR2_Mode (E_Reset);
```

9.3.23. DrvPMU_DetectVol

- **Prototype**

unsigned int DrvPMU_DetectVol (uBOR2TH)

- **Description**

Set BOR2 voltage

Configure the register 0x4040C[6 :4]=0.

- **Parameters**

uBOR2TH [in] :select BOR2 detect volyage.

0 : BOR2_17V 1 : BOR2_20V

2 : BOR2_22V 3 : BOR2_25V

4 : BOR2_27V 5 : BOR2_30V

6 : BOR2_36V 7 : BOR2_40V

- **Include**

Peripheral_lib/DrvPMU.h

© 2021-2022HYCON Technology Corp

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/*set BOR2 detect volyage 2.2V */
DrvPMU_DetectVol (BOR2_22V);
```

9.3.24. DrvPMU_ReadBOR2Status

- **Prototype**

  unsigned int DrvPMU_ReadBOR2Status(void)

- **Description**

  Read BOR2 status

  Read the register 0x4040C[2]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvPMU.h

- **Return Vaule**

  0：BOR2 status is High.

  1：BOR2 status is. (Low active)

- **Example**

```
/*read BOR2 status */
unsigned char Status;
Status= DrvPMU_ReadBOR2Status();
```

10. RTC Driver

10.1. Introduction

The following functions are included in RTC Manager Section.

Item	Functions	Description
01	DrvRTC_SetFrequencyCompensation	Set Frequency Compensation Data
02	DrvRTC_WriteEnable	Access Password to KEY to make access other register enable
03	DrvRTC_WriteDisable	Clear the RTC KEY to make the RTC register can not write
04	DrvRTC_ClockSource	Select clock source from the external or internal.
05	DrvRTC_AlarmEnable	Enable the TAEn.
06	DrvRTC_AlarmDisable	Disable the TAEn.
07	DrvRTC_PeriodicTimeEnable	Enable PTEn and set periodic timer frequency of RTC.
08	DrvRTC_PeriodicTimeDisable	Disable the PTEn.
09	DrvRTC_Enable	Enable the RTCEn.
10	DrvRTC_Disable	Disable the RTCEn.
11	DrvRTC_HourFormat	Set the clock for 12 or 24hour
12	DrvRTC_ReadState	Read the RTC state
13	DrvRTC_ClearState	Clear the RTC state
14	DrvRTC_EnableInt	RTC Interrupt Enable
15	DrvRTC_DisableInt	RTC Interrupt disable
16	DrvRTC_ReadIntFlag	Read the RTC Interrupt flag.
17	DrvRTC_ClearIntFlag	Clear the RTC Interrupt flag.
18	DrvRTC_Write	Set current date/time or alarm date/time to RTC
19	DrvRTC_Read	Read current date/time or alarm date/time from RTC setting

© 2021-2022HYCON Technology Corp

10.2. Type Definition

E_DRVRTC_CLOCK_SOURCE

Enumeration Identifier	Value	Description
E_EXTERNAL_CLOCK	2	RTC clock source from external low speed crystal source
E_INTERNAL_CLOCK	3	RTC clock source from internal low speed crystal source

E_DRVRTC_ TICK

Enumeration Identifier	Value	Description
E_DRVRTC_1_128_SEC	0	Set tick period 1/128 tick per second
E_DRVRTC_1_64_SEC	1	Set tick period 1/64 tick per second
E_DRVRTC_1_32_SEC	2	Set tick period 1/32 tick per second
E_DRVRTC_1_16_SEC	3	Set tick period 1/16 tick per second
E_DRVRTC_1_8_SEC	4	Set tick period 1/8 tick per second
E_DRVRTC_1_4_SEC	5	Set tick period 1/4 tick per second
E_DRVRTC_1_2_SEC	6	Set tick period 1/2 tick per second
E_DRVRTC_1_SEC	7	Set tick period 1 tick per second

E_DRVRTC_HOUR_FORMAT

Enumeration Identifier	Value	Description
E_DRVRTC_HOUR_12	1	The hour format by 12
E_DRVRTC_HOUR_24	0	The hour format by 24

E_DRVRTC_TIME_SELECT

Enumeration Identifier	Value	Description
DRVRTC_CURRENT_TIME	0	Select current time option
DRVRTC_ALARM_TIME	1	Select alarm time option

E_DRVRTC_FLAG

Enumeration Identifier	Value	Description
E_DRVRTC_ALARM_FLAG	0	alarm flag
E_DRVRTC_PERIODIC_FLAG	1	periodic timer flag
E_DRVRTC_CLEAR_ALL	2	clear alarm flag and periodic timer flag

© 2021-2022HYCON Technology Corp

10.3. Functions

Note : It is necessary to enable RTC clock and write <0110> in the RTKEY (register 0x41A00[23:20]) before writing data into the RTC register

10.3.1. DrvRTC_SetFrequencyCompensation

- **Prototype**

  unsigned int DrvRTC_SetFrequencyCompensation(

  unsigned int uFrequencyCom);

- **Description**

  Set Frequency Compensation Data

  Configure the register 0x41A04[22:16]

- **Parameters**

  uFrequencyCom [in] : specified RTC clock frequency compensation, It could be 0~0x7f

  0111111 : +126 ppm

  0111110 : +124 ppm

  |

  0000001 : +2 ppm

  0000000 : +0 ppm

  1000000 : - 0 ppm

  1000001 : - 2 ppm

  |

  1111110 : -124 ppm

  1111111 : -126 ppm

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Compensation deviation -2 PPM */

  DrvRTC_SetFrequencyCompensation(0x41);

10.3.2. DrvRTC_WriteEnable

- **Prototype**

  void DrvRTC_WriteEnable(void);

- **Description**

Access Password to KEY to make access other register enable.

Configure the register 0x41a00[23:20] =0110b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

- **Example**

```
/* Unlock RTC register, the RTC registers can be written */
DrvRTC_WriteEnable();
```

10.3.3. DrvRTC_WriteDisable

- **Prototype**

void DrvRTC_WriteDisable(void);

- **Description**

Clear the RTC KEY to make the RTC register can not write

Configure the register 0x41A00[23:20]=0000b

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

- **Example**

```
/* Lock RTC register, the RTC registers can not write */
DrvRTC_WriteDisable();
```

10.3.4. DrvRTC_ClockSource

- **Prototype**

unsigned int DrvRTC_ClockSource(uClockSource);

- **Description**

Select clock source from the external or internal.

Configure the register 0x40308[23:22]。

- **Parameters**

uClockSource [in]

0 : Disable

2 : E_EXTERNAL_CLOCK (LSXT enable, Otherwise, it is regarded as Disable)

3 : E_INTERNAL_CLOCK

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  Clock source configure result

- **Example**

```
/*Select the RTC Clock from external. */
DrvRTC_ClockSource(E_EXTERNAL_CLOCK);
```

10.3.5. DrvRTC_AlarmEnable

- **Prototype**

  void DrvRTC_AlarmEnable (void);

- **Description**

  Enable the RTC Alarm function.

  Configure the register0x41A00[3]=1.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  None

- **Example**

```
/* Enable RTC alarm*/
DrvRTC_AlarmEnable();
```

10.3.6. DrvRTC_AlarmDisable

- **Prototype**

  void DrvRTC_AlarmDisable (void);

- **Description**

  Disable the RTC Alarm function.

  Configure the register 0x41A00[3]=0.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

- **Example**

/* Disable Alarm */

DrvRTC_AlarmDisable();

10.3.7. DrvRTC_PeriodicTimeEnable

- **Prototype**

unsigned int DrvRTC_PeriodicTimeEnable (E_DRVRTC_TICK uPeriodicTimer);

- **Description**

Set periodic timer frequency of RTC.

Configure the register 0x41A04[2:0], 0x41A00[5]=1, 0x41A00[4]=1.

- **Parameters**

uPeriodicTimer[in] :

0: 1/128Second

1: 1/64Second

2: 1/32Second

3: 1/16Second

4: 1/8Second

5: 1/4Second

6: 1/2Second

7: 1Second

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Enable RTC alarm and select 1/16 second */

DrvRTC_PeriodicTimeEnable(3);

10.3.8. DrvRTC_PeriodicTimeDisable

- **Prototype**

void DrvRTC_PeriodicTimeDisable (void);

- **Description**

Disable the periodic time function.

Configure the register 0x41A00[5]=0 / 0x41A00[4]=0。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

- **Example**

/* Disable the PTEn */

DrvRTC_PeriodicTimeDisable();

10.3.9. DrvRTC_Enable

- **Prototype**

void DrvRTC_Enable (void);

- **Description**

Enable the RTC function.

Configure the register 0x41A00[0]=1。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

- **Example**

/* Enable the RTC */

DrvRTC_Enable();

10.3.10. DrvRTC_Disable

- **Prototype**

void DrvRTC_Disable (void);

- **Description**

Disable the RTC function.

Configure the register 0x41A00[0]=0.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

© 2021-2022HYCON Technology Corp

- **Example**

  /* Disable the RTC */

  DrvRTC_Disable();

10.3.11. DrvRTC_HourFormat

- **Prototype**

  unsigned int DrvRTC_HourFormat(E_DRVRTC_HOUR_FORMAT uHourFormat);

- **Description**

  Set the clock for 12 or 24hour

  Configure the register 0x41A00[2]。

- **Parameters**

  uHourFormat[in] : hour format

  0 : The hour format by 24

  1 : The hour format by 12

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

  /* Set 12-hour */

  DrvRTC_DrvRTC_HourFormat(1);

10.3.12. DrvRTC_ReadState

- **Prototype**

  unsigned int DrvRTC_ReadState(void);

- **Description**

  Read the RTC state

  Configure the register 0x41A00[19:16]

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  Return 0x0~0xf

  Bit 0 : RTC Alarm Flag

  Bit 1 : RTC Wakeup Flag

  Bit 2 : RTC Aeriodic Timer Flag

  Bit 3 : RTC Leap Year Flag

© 2021-2022HYCON Technology Corp

- **Example**

/* Check the RTC of alarm flag */

Sample code 1 :

If (DrvRTC_ReadState()&0x1)

//RTC alarm triggered

else

// RTC Wakeup triggered

Sample code 2 :

flag = DrvRTC_ReadState();

10.3.13. DrvRTC_ClearState

- **Prototype**

unsigned int DrvRTC_ClearState(E_DRVRTC_FLAG uFlag);

- **Description**

Clear the RTC state

Clear the register 0x41A00[19:16]

- **Parameters**

UFlag[in]:

0 : clear alarm flag

1 : clear periodic timer flag

2: clear alarm flag and periodic timer flag

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

/* Clear TAF/ PTF flag */

DrvRTC_ClearState(2);

10.3.14. DrvRTC_EnableInt

- **Prototype**

void DrvRTC_EnableInt(void)

- **Description**

RTC Interrupt Enable. Need to clear the PTF after response to interrupt,then it can response to interrupt normally next time.

Configure the register 0x40004[21]=1.

© 2021-2022HYCON Technology Corp

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  None

- **Example**

```
/* Enable RTC interrupt */
DrvRTC_EnableInt();
```

10.3.15. DrvRTC_DisableInt

- **Prototype**

  void DrvRTC_DisableInt(void)

- **Description**

  RTC Interrupt disable

  Configure the register 0x40004[21]=0.

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  None

- **Example**

```
/* Disable RTC interrupt */
DrvRTC_DisableInt();
```

10.3.16. DrvRTC_ReadIntFlag

- **Prototype**

  unsigned int DrvRTC_ReadIntFlag(void)

- **Description**

  Read the RTC Interrupt flag.

  Read the register 0x40004[5].

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvRTC.h

- **Return Vaule**

  0 : Normal

1 :Interrupted

- **Example**

/* Read the RTC Interrupt flag */

unsigned char flag；flag=DrvRTC_ReadIntFlag();

10.3.17. DrvRTC_ClearIntFlag

- **Prototype**

void DrvRTC_ClearIntFlag(void)

- **Description**

Clear the RTC Interrupt flag.

Configure the register 0x40004[5]=0.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

None

- **Example**

/* Clear the RTC Interrupt flag */

DrvRTC_ClearIntFlag();

10.3.18. DrvRTC_Write

- **Prototype**

unsigned int DrvRTC_Write (E_DRVRTC_TIME_SELECT eTime, S_DRVRTC_TIME_DATA_T *sPt);

- **Description**

Set current date/time or alarm date/time to RTC

Configure the register 0x41A08/0x41A0C/0x41A10/0x41A14/0x41A18/0x41A1C

- **Parameters**

eTime [in] : Specify the current/alarm time to be written.

0 : Current time

1 : Alarm time

*sPt [in] : Specify the data to write to RTC. It includes:

u8cClockDisplay	DRVRTC_CLOCK_12(00:00~11:59) / DRVRTC_CLOCK_24(00:00~23:59)
u8cAmPm	DRVRTC_AM / DRVRTC_PM
u32cSecond	Second value
u32cMinute	Minute value
u32cHour	Hour value
u32cDayOfWeek	Day of week
u32cDay	Day value

© 2021-2022HYCON Technology Corp

u32cMonth Month value

u32Year Year value

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Condition: Update current the second of time to zero */
S_DRVRTC_TIME_DATA_T sCurTime;
DrvRTC_Read(DRVRTC_ALARM_TIME, &sCurTime);
sCurTime.u32cSecond = 0;
DrvRTC_Write(DRVRTC_ALARM_TIME, &sCurTime);
```

10.3.19. DrvRTC_Read

- **Prototype**

```
unsigned int DrvRTC_Read (
E_DRVRTC_TIME_SELECT eTime,
S_DRVRTC_TIME_DATA_T *sPt );
```

- **Description**

Read current date/time or alarm date/time from RTC setting

Configure the register 0x41A08/0x41A0C/0x41A10/0x41A14/0x41A18/0x41A1C

- **Parameters**

eTime [in] : Specify the current/alarm time to be written.

0 : Current time

1 : Alarm time

*sPt [in] : Specify the data to write to RTC. It includes:

u8cClockDisplay DRVRTC_CLOCK_12 / DRVRTC_CLOCK_24

u8cAmPm DRVRTC_AM / DRVRTC_PM

u32cSecond Second value

u32cMinute Minute value

u32cHour Hour value

u32cDayOfWeek Day of week

u32cDay Day value

u32cMonth Month value

u32Year Year value

- **Include**

Peripheral_lib/DrvRTC.h

- **Return Vaule**

0: Operation successful

Other : Incorrect argument

- **Example**

```
/* Condition: You want to get current RTC calendar and time */
S_DRVRTC_TIME_DATA_T sCurTime;
DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);
```

11. I2C Driver

11.1. Introduction

The following functions are included in I2C Manager Section.

Item	Functions	Description
01	DrvI2C_Open	Enable the I2C and configure the I2C bus clock
02	DrvI2C_Close	Disable the I2C
03	DrvI2C_SlaveSet	Enable slave mode，set address，choose whether to enable GC
04	DrvI2C_SlaveDisable	Disable Slave mode
05	DrvI2C_SetIOPin	Select IO port as IIC port
06	DrvI2C_WriteData	To set a byte of data to be sent.
07	DrvI2C_Write3ByteData	To set a 3byte of data to be sent.
08	DrvI2C_ReadData	Read the data form receiver data buffer.
09	DrvI2C_Ctrl	To set I2C control bit include STA, STO, AA, SI in control register.
10	DrvI2C_EnableInt	Enable the I2C Interrupt
11	DrvI2C_DisableInt	Disable the I2C Interrupt
12	DrvI2C_ReadIntFlag	Read the I2C Interrupt flag.
13	DrvI2C_ClearIntFlag	Clear the I2C Interrupt flag.
14	DrvI2C_ClearEIRQ	Clear the EIRQ
15	DrvI2C_ClearIRQ	Clear the IRQ.
16	DrvI2C_GetStatusFlag	Take status flags
17	DrvI2C_TimeOutEnable	Enable TimeOut，set clock pre scale and time out limit
18	DrvI2C_TimeOutDisable	Disable the Timeout
19	DrvI2C_STSP	Generate the START or STOP singal from IIC bus
20	DrvI2C_MGetACK	Check the ACK from slaver during the set time
21	DrvI2C_DisableIOPin	Disable IIC communication function of the IO port
22	DrvI2C_Reset	Restart the I2C

© 2021-2022HYCON Technology Corp

11.2. Type Definition

E_DRVI2C_Status

Enumeration Identifier	Value	Description
E_DRVI2C_ARBITRATION_FLAG	0	Arbitration Lost Flag
E_DRVI2C_GENERAL_CALL_FLAG	1	General Call Flag
E_DRVI2C_ACKNOWLEDGE_FLAG	2	Acknowledge Flag
E_DRVI2C_DATA_FIELD_FLAG	3	Data Field Flag
E_DRVI2C_RW_STATE_FLAG	4	Read/Write State Flag
E_DRVI2C_RS_FLAG	5	Received Stop/Repeat-Start Flag
E_DRVI2C_SLAVE_ACTIVE_FLAG	6	Slave Mode Active Flag
E_DRVI2C_MASTER_ACTIVE_FLAG	7	Master Mode Active Flag

E_DRVI2C_TIMEOUT_PRESCALE

Enumeration Identifier	Value	Description
E_DRVI2C_I2CLK_DIV_1	0	I2C CLK/1
E_DRVI2C_I2CLK_DIV_2	1	I2C CLK/2
E_DRVI2C_I2CLK_DIV_4	2	I2C CLK/4
E_DRVI2C_I2CLK_DIV_8	3	I2C CLK/8
E_DRVI2C_I2CLK_DIV_16	4	I2C CLK/16
E_DRVI2C_I2CLK_DIV_32	5	I2C CLK/32
E_DRVI2C_I2CLK_DIV_64	6	I2C CLK/64
E_DRVI2C_I2CLK_DIV_128	7	I2C CLK/128

E_DRVI2C_TIMEOUT_LIMIT

Enumeration Identifier	Value	Description
E_DRVI2C_CLKPSX1	0	1 * CLKps Cycle
E_DRVI2C_CLKPSX2	1	2 * CLKps Cycle
E_DRVI2C_CLKPSX3	2	3 * CLKps Cycle
E_DRVI2C_CLKPSX4	3	4 * CLKps Cycle
E_DRVI2C_CLKPSX5	4	5 * CLKps Cycle
E_DRVI2C_CLKPSX6	5	6 * CLKps Cycle
E_DRVI2C_CLKPSX7	6	7 * CLKps Cycle
E_DRVI2C_CLKPSX8	7	8 * CLKps Cycle
E_DRVI2C_CLKPSX9	8	9 * CLKps Cycle
E_DRVI2C_CLKPSX10	9	10 * CLKps Cycle
E_DRVI2C_CLKPSX11	10	11 * CLKps Cycle
E_DRVI2C_CLKPSX12	11	12 * CLKps Cycle
E_DRVI2C_CLKPSX13	12	13 * CLKps Cycle
E_DRVI2C_CLKPSX14	13	14 * CLKps Cycle
E_DRVI2C_CLKPSX15	14	15 * CLKps Cycle
E_DRVI2C_CLKPSX16	15	16 * CLKps Cycle

E_DRVI2C_INTERRUPT

Enumeration Identifier	Value	Description
E_DRVI2C_INT	1	I2C Interrupt enable
E_DRVI2C_ERROR_INT	2	I2C error Interrupt enable
E_DRVI2C_INT_ALL	3	enable I2C interrupt and error interrupt

E_DRVI2C_SLAVE_BIT

Enumeration Identifier	Value	Description
E_DRVI2C_SLAVE_7BIT	0	Slave 7bit address mode
E_DRVI2C_SLAVE_10BIT	1	Slave 10bit address mode

© 2021-2022HYCON Technology Corp

11.3. Functions

Note：Configure the IIC register after enable IIC

11.3.1. DrvI2C_Open

- **Prototype**

  unsigned int DrvI2C_Open (uint32_t u32CRG);

- **Description**

  Enable the I2C and configure the I2C bus clock

  Configure the register 0x41000[0]=1, 0x41008[23:16]

- **Parameters**

  u32CRG [in] : Set CRG value. It could be 0~0xff.

  Data Baud Rate = (I2CLK/(4*(CRG+1)))

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Vaule**

  0: Operation successful

  Other : Incorrect argument

- **Example**

```
/* Enable I2C and set CRG value 100 */
DrvI2C_Open(100);
```

11.3.2. DrvI2C_Close

- **Prototype**

  void DrvI2C_Close (void);

- **Description**

  Disable the I2C.

  Configure the register 0x41000[0]=0。

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Vaule**

  None

- **Example**

```
/* Close the I2C */
DrvI2C_Close();
```

© 2021-2022HYCON Technology Corp

11.3.3. DrvI2C_SlaveSet

- **Prototype**

```
unsigned int DrvI2C_SlaveSet( uint32_t uSlaveAddr,
                              E_DRVI2C_SLAVE_BIT uAddrBit,
                              uint8_t uSlave3Byte,
                              uint8_t GC_Flag );
```

- **Description**

Enable slave mode, and set the location address, and choose whether to enable GC

Configure the register 0x41004[7] /0x41004[5] /0x41000[2] /0x4100C[7:0]

- **Parameters**

uSlaveAddr : slaver address

7bit : 0~0x7f

10bit: 0~0x3ff

uAddrBit : slaver address mode

0: Slave 7bit address mode

1: Slave 10bit address mode

uSlave3Byte: Slave 3 Byte Data Mode Enable control

0: Normal

1:Slave 3byte Data transfer

GC_Flag : general call flag

0: Normal

1: Enable general call

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

1: Operation successful

Other : Incorrect argument

- **Example**

```
/* Enable Slave mode, position setting 0x30*/
DrvI2C_SlaveSet(0x30,0,0,0);
```

11.3.4. DrvI2C_SlaveDisable

- **Prototype**

```
void DrvI2C_SlaveDisable(void);
```

- **Description**

Disable I2C slave mode.

Configure the register 0x41004[7]=0。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* disable I2C slave mode */
DrvI2C_SlaveDisable();
```

11.3.5. DrvI2C_SetIOPin

- **Prototype**

unsigned char DrvI2C_SetIOPin(unsigned int upin);

- **Description**

Set IO port of I2C.

Configure the register 0x40844[19 :16] 。

- **Parameters**

upin[in] : select IO port as I2C port

0	SCL=PT1.0;SDA=PT1.1
1	SCL=PT1.2;SDA=PT1.3
2	SCL=PT1.4;SDA=PT1.5
3	SCL=PT1.6;SDA=PT1.7
4	SCL=PT2.0;SDA=PT2.1
5	SCL=PT2.2;SDA=PT2.3
6	SCL=PT2.4;SDA=PT2.5
7	SCL=PT2.6;SDA=PT2.7

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* specify PT1.0 and PT1.1 */
DrvI2C_SetIOPin(0);
```

11.3.6. DrvI2C_WriteData

- **Prototype**

void DrvI2C_WriteData(uint8_t uData);

- **Description**

To set a byte of data to be sent.

© 2021-2022HYCON Technology Corp

Configure the register 0x41014[7:0].

- **Parameters**

uData [IN] : the data to be sent

1 Byte data

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

/* Set byte data 0x55 into transmitter data buffer register */

DrvI2C_WriteData(0x55);

11.3.7. DrvI2C_Write3ByteData

- **Prototype**

void DrvI2C_Write3ByteData(uint8_t uData1,uData2,uData3);

- **Description**

To set a 3byte of data to be sent.

Configure the register.

- **Parameters**

uData1, uData2, uData3 [IN] : Byte data. Input range is : 0~0xFF

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

/* Set 3byte data 0x11 0x22 0x33 into transmitter dada buffer register */

DrvI2C_Write3ByteData(0x11,0x22,0x33);

11.3.8. DrvI2C_ReadData

- **Prototype**

unsigned char DrvI2C_ReadData(void);

- **Description**

Read the data form receiver data buffer.

Configure the register 0x41010[7:0]的值。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

1 Byte received

- **Example**

/* Read byte data from receiver data buffer */

unsigned int data; data=DrvI2C_ReadData();

11.3.9. DrvI2C_Ctrl

- **Prototype**

void DrvI2C_Ctrl(uint8_t start, uint8_t stop, uint8_t intFlag, uint8_t ack);

- **Description**

To set I2C control bit include STA, STO, AA, SI in control register.

Configure the register 0x41004[3:0]

- **Parameters**

start [in]:

To set STA bit or not. (1: set, 0: don"t set). If the STA bit is set, a START or repeat START signal will be generated when I2C bus is free.

stop [in]:

To set STO bit or not. (1: set, 0: don"t set). If the STO bit is set, a STOP signal will be generated. When a STOP condition is detected, this bit will be cleared by hardward automatically.

intFlag [in]:

To clear SI flag (I2C interrupt flag). (1: clear, 0: don"t work)

ack [in]:

To enable AA bit (Assert Acknowledge control bit) or not. (1: enable, 0: disable)

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

DrvI2C_Ctrl(0, 0, 1, 0); /* Set I2C SI bit to clear SI flag */

DrvI2C_Ctrl(1, 0, 0, 0); /* Set I2C STA bit to send START signal */

11.3.10. DrvI2C_EnableInt

- **Prototype**

void DrvI2C_EnableInt(E_DRVI2C_INTERRUPT uINT)

- **Description**

Enable the I2C Interrupt

Configure the register 0x40000[21:20]

- **Parameters**

uINT[IN]

0 : I2C Interrupt enable

1 : I2C error Interrupt enable

2 : enable I2C interrupt and error interrupt

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Enable I2C interrupt */
DrvI2C_EnableInt(1);
```

11.3.11. DrvI2C_DisableInt

- **Prototype**

void DrvI2C_DisableInt(E_DRVI2C_INTERRUPT uINT)

- **Description**

Disable the I2C Interrupt

Configure the register 0x40000[21:20]

- **Parameters**

uINT[IN] :

0: I2C Interrupt disable

1 : I2C error Interrupt disable

2 : disable I2C interrupt and error interrupt

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Disable I2C interrupt */
DrvI2C_DisableInt(1);
```

11.3.12. DrvI2C_ReadIntFlag

- **Prototype**

E_DRVI2C_INTERRUPT DrvI2C_ReadIntFlag(void)

- **Description**

Read the I2C Interrupt flag.

Read the register 0x40000[5:4].

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

0: no I2C IRQ

1 : I2C Interrupt flag is true

2 : I2C error Interrupt flag is true

3 : enable I2C interrupt and error interrupt of flag is true

- **Example**

```
/* Read the I2C Interrupt flag */
uint32_t temp;
temp=DrvRTC_ReadIntFlag();
```

11.3.13. DrvI2C_ClearIntFlag

- **Prototype**

void DrvI2C_ClearIntFlag(E_DRVI2C_INTERRUPT uINT)

- **Description**

Clear the RTC Interrupt flag.

Clear the register 0x40000[5:4].

- **Parameters**

uINT[IN] :

0 : Clear the I2C Interrupt flag

1 : Clear the I2C error Interrupt flag

2 : Clear the I2C interrupt and error flag

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Clear the I2C Interrupt flag    */
DrvI2C_ClearIntFlag(0);
```

11.3.14. DrvI2C_ClearEIRQ

- **Prototype**

void DrvI2C_ClearEIRQ(void)

- **Description**

Clear the EIRQ.

Note :The EIRQ flag can be clear after clear the TOPFLAG. The I2CEIF can be clear after clear the EIRQ

flag. Write 0 to this bit to clear EIRQ.

Configure the register 0x41004[4]=0。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Clear the EIRQ */
DrvI2C_ClearEIRQ();
```

11.3.15. DrvI2C_ClearIRQ

- **Prototype**

void DrvI2C_ClearIRQ(void)

- **Description**

Clear the IRQ.

The IRQFlag will be set 1 when receive 9th clock, and SCL will be pull down until clear IRQFlag

Configure the register 0x41004[1]=0 。

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Clear the IRQ  */
DrvI2C_ClearIRQ();
```

11.3.16. DrvI2C_GetStatusFlag

- **Prototype**

unsigned char DrvI2C_GetStatusFlag(void)

- **Description**

Take status flags

Read the register 0x41004[23:16].

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

The meaning of each bit for the return value:

Bit 0 : Arbitration Lost Flag

Bit 1: General Call Flag

Bit 2: Acknowledge Flag

Bit 3: Data Field Flag

Bit 4: Read/Write State Flag

Bit 5: Received Stop/Repeat-Start Flag

Bit 6: Slave Mode Active Flag

Bit 7: Master Mode Active Flag

ARB:uStatus=0

0 : Normal

1 : Arbitration Lost

GC:uStatus=1

0 : Normal

1 : Currently General Call Operation

A/NA:uStatus=2

0 : No Ack has been transmitted or received.

1 : Ack has been transmitted or received.

DF:uStatus=3

0 : Normal

1 : I2C Data Byte has been transmitted or received.

R/W:uStatus=4

0 : Write Command has been transmitted or received.

1 : Read Command has been transmitted or received.

RX P/Sr: uStatus=5

0 : Normal

1 : Stop/Repeat-Start has been transmitted or received.

SAct:uStatus=6

0 : Inactive

1 : Active

MAct: uStatus=7

0 : Inactive

1 : Active

- **Example**

```
/* 讀取應答信號標誌位元 /
DrvRTC_GetStatusFlag(2);   //讀取應答信號ACK的狀態標誌位元
```

© 2021-2022HYCON Technology Corp

11.3.17. DrvI2C_TimeOutEnable

- **Prototype**

```
unsigned char DrvI2C_TimeOutEnable(
E_DRVI2C_TIMEOUT_PRESCALE uPreScale,
E_DRVI2C_TIMEOUT_LIMIT uTimeOutLimit
);
```

- **Description**

Enable TimeOut, and set the clock pre scale and time out limit

Configure the register 0x41000[1]=1, 0x41008[6:0].

- **Parameters**

uPreScale[in]:

0	I2C CLK/1
1	I2C CLK/2
2	I2C CLK/4
3	I2C CLK/8
4	I2C CLK/16
5	I2C CLK/32
6	I2C CLK/64
7	I2C CLK/128

uTimeOutLimit [in] :

0	1 * CLKps Cycle
1	2 * CLKps Cycle
2	3 * CLKps Cycle
3	4 * CLKps Cycle
4	5 * CLKps Cycle
5	6 * CLKps Cycle
6	7 * CLKps Cycle
7	8 * CLKps Cycle
8	9 * CLKps Cycle
9	10 * CLKps Cycle
10	11 * CLKps Cycle
11	12 * CLKps Cycle
12	13 * CLKps Cycle
13	14 * CLKps Cycle
14	15 * CLKps Cycle
15	16 * CLKps Cycle

- **Include**

© 2021-2022HYCON Technology Corp

Peripheral_lib/DrvI2C.h

- **Return Vaule**

0: Operation successful

0xff: Incorrect argument

- **Example**

```
/* Enable TimeOut，set clock pre scale / 32 and time out limit=15 * CLKps Cycle   */
DrvI2C_TimeOutEnable(5,14);
```

11.3.18. DrvI2C_TimeOutDisable

- **Prototype**

void DrvI2C_TimeOutDisable(void)

- **Description**

Disable the Timeout

Configure the register 0x41000[1] =0.

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Disable time out   */
DrvI2C_TimeOutDisable();
```

11.3.19. DrvI2C_STSP

- **Prototype**

void DrvI2C_STSP(unsigned char usignal);

- **Description**

Generate the START or STOP singal from IIC bus.

Configure the register 0x41004[3:2].

- **Parameters**

usignal[in] : singal control

0 Generate START singal

1 Generate STOP singal

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

© 2021-2022HYCON Technology Corp

- **Example**

```
/* Generate the START or STOP singal*/
DrvI2C_STSP(0);
```

11.3.20. DrvI2C_MGetACK

- **Prototype**

```
unsigned char DrvI2C_MGetACK(unsigned int utime);
```

- **Description**

Check the ACK from slaver during the set time

Configure the register 0x41004[1]

- **Parameters**

utimel[in]

the set time :0~0xffff

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

0: ACK was returned

1: No ACK was returned

- **Example**

```
/* check the ACK during the 0xffff time*/
Err_flag=DrvI2C_MGetACK(0xffff);
```

11.3.21. DrvI2C_DisableIOPin

- **Prototype**

```
void DrvI2C_DisableIOPin(void)
```

- **Description**

Disable IIC communication function of the IO port

Configure the register 0x40844[16]=0

- **Parameters**

None

- **Include**

Peripheral_lib/DrvI2C.h

- **Return Vaule**

None

- **Example**

```
/* Disable IIC communication function of the IO port    */
DrvI2C_DisableIOPin();
```

© 2021-2022HYCON Technology Corp

11.3.22. DrvI2C_Reset

- **Prototype**

  void DrvI2C_Reset (void)

- **Description**

  Restart the I2C

  Configure the register 0x41000[0], 0x41004[4:0]

  Turn off the I2C and clear the error flag, and then turn on the I2C function to prevent the I2C from continuing to be in an abnormal state。

- **Parameters**

  None

- **Include**

  Peripheral_lib/DrvI2C.h

- **Return Vaule**

  None

- **Example**

```
/*Restart I2C*/
DrvI2C_Reset ();
```

12. LCD Driver

12.1. Introduction

The following functions are included in LCD Manager Section

Item	Functions	Description
01	DrvLCD_EnableCLK	Set LCD clock source
02	DrvLCD_DisplayMode	Set LCD display mode
03	DrvLCD_VLCDMode	Set VLCD bias voltage
04	DrvLCD_LcdDuty	Set LCD operation period
05	DrvLCD_LCDBuffer	Set LCD buffer
06	DrvLCD_LCDEnable	Set LCD enable(LCD clock output to SEG/COM Port or not)
07	DrvLCD_VLCDEnable	Set VLCD Pump
08	DrvLCD_LCDBias	Set LCD bias
09	DrvLCD_IOMode	Select the operation mode of PT6~PT10 and COM5/COM4
10	DrvLCD_WriteData	Write data to LCD data buffer(LCD0~LCD17)

© 2021-2022HYCON Technology Corp

12.2. Type Definition

E_VLCD_MODE

Enumeration Identifier	Value	Description
E_VLCD28	2	VLCD=2.8V
E_VLCD30	3	VLCD=3.0V
E_VLCD33	4	VLCD=3.3V
E_VLCD39	5	VLCD=3.9V
E_VLCD45	6	VLCD=4.5V
E_VLCD50	7	VLCD=5.0V

E_LCD_DUTY

Enumeration Identifier	Value	Description
E_LCD_DUTY3	0	LCD operation period: 1/3 duty
E_LCD_DUTY4	1	LCD operation period: 1/4 duty
E_LCD_DUTY5	2	LCD operation period: 1/5 duty
E_LCD_DUTY6	3	LCD operation period: 1/6 duty
E_LCD_DUTY7	4	LCD operation period: 1/7 duty
E_LCD_DUTY8	5	LCD operation period: 1/8 duty

E_LCD_DISPLAY_MDE

Enumeration Identifier	Value	Description
E_LCD_NORMAL	0	Normal mode
E_LCD_PIXELON	1	The LCD is turned on no matter what the input is
E_LCD_PIXELOFF	2	The LCD is turned off no matter what the input is

© 2021-2022HYCON Technology Corp

12.3. Functions

12.3.1. DrvLCD_EnableCLK

- **Prototype**

  unsigned char DrvLCD_EnableCLK(unsigned int uLCD1,unsigned int uLCD2,unsigned int usource)

- **Description**

  Set LCD clock source. Select frequency divider of LCDE/LCDO clock source

  Configure the register 0x40310[6:0]

- **Parameters**

  uLCD1[in] : Select frequency divider of LCDO

  0 : ÷ 1; 1 : ÷ 3; 2 : ÷ 5; 3 : ÷ 7

  4 : ÷ 9; 5 : ÷ 11; 6 : ÷ 13; 7 : ÷ 15

  uLCD2[in] : Select frequency divider of LCDE

  0 : Disable; 1 : ÷ 1; 2 : ÷ 2; 3 : ÷ 4

  4 : ÷ 8; 5 : ÷ 16; 6 : ÷ 32; 7 : Disable

  usource[in] : LCD clock source selection

  0 : LS_CK(alawys/8)

  1 : HS_CK(alawys/64)

- **Include**

  Peripheral_lib/DrvLCD.h

- **Return Vaule**

  0: Operation successful

  1 : Incorrect argument

- **Example**

```
/* set HS_CK as LCD clock source, and the frequency divider LCD1*LCD2=5*1 */
DrvLCD_EnableCLK(2,1,1);
```

12.3.2. DrvLCD_DisplayMode

- **Prototype**

  unsigned char DrvLCD_DisplayMode(unsigned int uDISMODE)

- **Description**

  Set LCD display mode.

  Configure the register 0x41B00[17:16]

- **Parameters**

  uDISMODE[in] : Set LCD display mode

  0 : normal mode

  1 : The LCD is turned on no matter what the input is

© 2021-2022HYCON Technology Corp

2： The LCD is turned off no matter what the input is.

- **Include**

Peripheral_lib/DrvLCD.h

- **Return Vaule**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/* set as normal mode */
DrvLCD_DisplayMode(E_LCD_NORMAL);
```

12.3.3. DrvLCD_VLCDMode

- **Prototype**

unsigned char DrvLCD_VLCDMode(unsigned int uVLCDMODE)

- **Description**

Set VLCD bias voltage.

Configure the register 0x41B00[2:0]

- **Parameters**

uVLCDMODE[in] : LCD bias voltage selection

2：2.8V, Charge PUMP on, VLCD R off

3：3.0V, Charge PUMP on, VLCD R off

4：3.3V, Charge PUMP on, VLCD R off

5：3.9V, Charge PUMP on, VLCD R off

6：4.5V, Charge PUMP on, VLCD R off

7：5.0V, Charge PUMP on, VLCD R off

- **Include**

Peripheral_lib/DrvLCD.h

- **Return Vaule**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/* set 3.0V as LCD bias voltage  */
DrvLCD_VLCDMode(E_VLCD30);
```

12.3.4. DrvLCD_LcdDuty

- **Prototype**

unsigned char DrvLCD_LcdDuty(unsigned int uDUTY)

- **Description**

Set LCD operation period.

Configure the register 0x41B00[5:4]

- **Parameters**

  uDUTY[in] : LCD operation period selection

  0：1/3 Duty　　1：1/4 Duty

  2：1/5 Duty　　3：1/6 Duty

  4：1/7 Duty　　5：1/8 Duty

- **Include**

  Peripheral_lib/DrvLCD.h

- **Return Vaule**

  0: Operation successful

  1 : Incorrect argument

- **Example**

```
/* set as 1/4 Duty */
DrvLCD_LcdDuty(E_LCD_DUTY4);
```

12.3.5. DrvLCD_LCDBuffer

- **Prototype**

  unsigned char DrvLCD_LCDBuffer(unsigned int uBEN)

- **Description**

  Set VLCD buffer.

  Configure the register 0x41B00[3]

- **Parameters**

  uBEN[in] : VLCD buffer control

  0 : disable

  1 : enable

- **Include**

  Peripheral_lib/DrvLCD.h

- **Return Vaule**

  0: Operation successful

  1 : Incorrect argument

- **Example**

```
/* enable VLCD buffer   */
DrvLCD_LCDBuffer(1);
```

12.3.6. DrvLCD_LCDEnable

- **Prototype**

  unsigned char DrvLCD_LCDEnable(unsigned int umode);

- **Description**

  Set LCD Clock output to SEG/COM Port or not.

© 2021-2022HYCON Technology Corp

Configure the register 0x41B00[7]/0x41B00[7]

- **Parameters**

umode [in] :

0 : close LCD clock

1 : enable LCD clock

- **Include**

Peripheral_lib/DrvLCD.h

- **Return Vaule**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/* Enable LCD */
DrvLCD_LCDEnable(1);
```

12.3.7. DrvLCD_VLCDEnable

- **Prototype**

unsigned char DrvLCD_VLCDEnable(unsigned int umode);

- **Description**

Set VLCD bias voltage.

Configure the register 0x41B00[19]

- **Parameters**

umode [in] : VLCD Pump

0 : VLCD Punp OFF, Can be input by external voltage, R-Type

1 : VLCD Punp ON

- **Include**

Peripheral_lib/DrvLCD.h

- **Return Vaule**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/*enable VLCD pump    */
DrvLCD_VLCDEnable(1);
```

12.3.8. DrvLCD_LCDBias

- **Prototype**

unsigned char DrvLCD_LCDBias(E_LCD_BIAS ubias);

- **Description**

Set LCD bias

Configure the register 0x41B00[21]

- **Parameters**

ubias [in] :

E_LCD_BIAS3：1/3 Bias

E_LCD_BIAS4：1/4 Bias

- **Include**

Peripheral_lib/DrvLCD.h

- **Return Vaule**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/*set LCD 1/3 Bias */
DrvLCD_LCDBias(E_LCD_BIAS3);
```

12.3.9. DrvLCD_IOMode

- **Prototype**

unsigned char DrvLCD_IOMode(unsigned int uport,unsigned int uIOMODE)

- **Description**

Select the operation mode of PT6~PT13 and COM5/COM4.

Configure the register 0x41B04/0x41B08.

- **Parameters**

uport[in] : specified port

0：PT6　　1：PT7　　2：PT8

3：PT9　　4：PT10　　5：PT13

uIOMODE[in]　: It could be 0~0xff.

The each bit of uIOMODE stand for the mode of corresponding pin

0：I/O mode

1：LCD mode

- **Include**

Peripheral_lib/DrvLCD.h

- **Return Vaule**

0: Operation successful

1 : Incorrect argument

- **Example**

```
/* set PT6 is LCD Mode * /
DrvLCD_IOMode(E_LCD_PT6LEN,0xFF);
```

© 2021-2022HYCON Technology Corp

12.3.10. DrvLCD_WriteData

● **Prototype**

unsigned char DrvLCD_WriteData(unsigned int uSEG,unsigned int data)

● **Description**

Write data to LCD data buffer(LCD0~LCD17)

Configure the register 0x40850~0x40894.

● **Parameters**

uSEG[in] : LCD Data Buffer(LCD0~LCD17)

Each buffer has two SEG,such as LCD0=SEG1:SEG0;

0~21: corresponding to LCD0~LCD21

LCD0=SEG1:SEG0; //0x408C8

LCD1=SEG3:SEG2; //0x40850

LCD2=SEG5:SEG4; //0x40854

LCD3=SEG7:SEG6; //0x40858

LCD4=SEG9:SEG8; //0x4085C

LCD5=SEG11:SEG10; //0x40860

LCD6=SEG13:SEG12; //0x40864

LCD7=SEG15:SEG14; //0x40868

LCD8=SEG17:SEG16; //0x4086C

LCD9=SEG19:SEG18; //0x40870

LCD10=SEG21:SEG20; //0x40874

LCD11=SEG23:SEG22; //0x40878

LCD12=SEG25:SEG24; //0x4087C

LCD13=SEG27:SEG26; //0x40880

LCD14=SEG29:SEG28; //0x40884

LCD15=SEG31:SEG30; //0x40888

LCD16=SEG33:SEG32; //0x4088C

LCD17=SEG35:SEG34; //0x40890

LCD18=SEG37:SEG36; //0x40894

LCD19=SEG39:SEG38; //0x40898

LCD20=SEG41:SEG40; //0x4089C

LCD21=SEG43:SEG42; //0x408CC

Data[in] : the data written to LCD buffer, the data is adapted to our SEG position arrangement,it could be 0~0xfff;

Note :

The data needs to configure your own LCD panel and SEG line arrangement is in agreement.

The data be written to LCD buffer, should be noted the LCD Duty and data format setting.

EX : LCD Duty=1/6 Duty, write data to LCD1. The data format : data[5:0]=SEG3, data[11:6]=SEG2

© 2021-2022HYCON Technology Corp

- **Include**

  Peripheral_lib/DrvLCD.h

- **Return Vaule**

  0: Operation successful

  1 : Incorrect argument

- **Example**

```
/* Set LCD duty=1/6Duty, and write data to LCD1 */
DrvLCD_LcdDuty (E_LCD_DUTY6);
DrvLCD_WriteData(1,0x03F);   //0x40850=0x003f0000
DrvLCD_WriteData(1,0xFC0);   //0x40850=0x0000003f
DrvLCD_WriteData(1,0xFFF);   //0x40850=0x003f003f
/* Set LCD duty=1/4Duty, and write data to LCD1 */
DrvLCD_LcdDuty (E_LCD_DUTY4);
DrvLCD_WriteData(1,0x03F);   //0x40850=0x000f0003
DrvLCD_WriteData(1,0xFF);   //0x40850=0x000f000f
DrvLCD_WriteData(1,0xF0);   //0x40850=0x0000000f
```

© 2021-2022HYCON Technology Corp

13. FLASH Read/Write Driver

13.1. Introduction

The following functions are included in FLASH Manager Section.

Item	Functions	Description
01	ISP_FUNC_ROMP->FlashOpEn	Enable Flash operate
02	ISP_FUNC_ROMP->FlashOpDis	Disable Flash operate
03	ISP_FUNC_ROMP->Burn_Word	Write a data of word to the specified address
04	ISP_FUNC_ROMP->BurnPage	Write 32 data of word to the specified address in a row
05	ReadWord	Read a data of word from the specified address
06	ReadPage	Read 32 data of word from the specified address in a row
07	ISP_FUNC_ROMP->SectorErase	Erase one sector to the specified address
08	ISP_FUNC_ROMP->CRC	calculate CRC
09	ISP_FUNC_ROMP->fastBlank	Fast Blank Check

© 2021-2022HYCON Technology Corp

13.2. Functions

Note1 : User has to do SYS_DisableGIE, before execute Flash burn/read function. Disable system global GIE function, it can prevent program exception when executing Flash burn/read function.

Note2 : VDD5V have to more than 2V, it can prevent program burn error when executing Flash burn function.

13.2.1. ISP_FUNC_ROMP->FlashOpEn

- **Prototype**

  int ISP_FUNC_ROMP->FlashOpEn(void);

- **Description**

  Release FLASH operate protection, and than FLASH can be write normally.

- **Parameters**

  None

- **Include**

  Peripheral_lib/Drvflash.h

- **Return Vaule**

  0

- **Example**

```
/* Release FLASH operate protection */
ISP_FUNC_ROMP->FlashOpEn();
```

13.2.2. ISP_FUNC_ROMP->FlashOpDis

- **Prototype**

  int ISP_FUNC_ROMP->FlashOpDis(void);

- **Description**

  Enable FLASH operate protection.Can’t write FLASH

- **Parameters**

  None

- **Include**

  Peripheral_lib/Drvflash.h

- **Return Vaule**

  0

- **Example**

```
/* Enable FLASH operate protection */
ISP_FUNC_ROMP->FlashOpDis();
```

© 2021-2022HYCON Technology Corp

13.2.3. ISP_FUNC_ROMP->Burn_Word

- **Prototype**

  int ISP_FUNC_ROMP->BurnWord(uint32_t addr, uint32_t data);

- **Description**

  Write a data of word to the specified address.

  Operating time about 20us.

- **Parameters**

  addr[in] : the address to be written

  The input range is 0~0xffff, and the start address of flash is 0x90000. The interval of address is 4 bytes.

  For exemple：The address of 0x9a880 will written a word if addr[in]=0x9a880.

  Delay time[in] : delay time of burning

  data [in] : the data to be written, it could be 0~0xffffffff

- **Include**

  Peripheral_lib/Drvflash.h

- **Return Vaule**

  0 : successful

  3 : address error

- **Example**

```
/* write the data of 0xFF05 to address of 0x90880 */
ISP_FUNC_ROMP->FlashOpEn();
ISP_FUNC_ROMP->BurnWord(0x90880, 0xFF05);
ISP_FUNC_ROMP->FlashOpDis();
```

  Note : VDD5V have to more than 1.8V, it can prevent program burn error when executing Flash burn function.

13.2.4. ISP_FUNC_ROMP->BurnPage

- **Prototype**

  int ISP_FUNC_ROMP->BurnPage(uint32_t addt, uint32_t *data, int len);

- **Description**

  Write 32 data of word to the specified address in a row one time.

  Operating time about 640us.

- **Parameters**

  addr[in] : the initial address to be written

  The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 128(32*4) bytes, and the page only can be written one by one, for only 128byte in each page . and the address only could be 0xuu00 or 0xuu80. (u is defined by user)

  For exemple：The address of 0x9a880 will written a word if addr[in]=0xa880.

  Delay time[in] : delay time of burning

  data [in] : the data to be written

© 2021-2022HYCON Technology Corp

it could be 0~0xffffffff . The length of data[in] is 32 word

- **Include**

Peripheral_lib/Drvflash.h

- **Return Vaule**

0 : successful

3 : wrong address

4 : address not multiple of 4

5 : wrong length

- **Example**

```
/* write 32 data of word to address of 0x90880 in a row one time */
unsigned int *A[32]={0};
ISP_FUNC_ROMP->FlashOpEn();
ISP_FUNC_ROMP->BurnPage(0x90880, A, 32);
ISP_FUNC_ROMP->FlashOpDis();
```

Note : VDD5V have to more than 1.8V, it can prevent program burn error when executing Flash burn function.

13.2.5. ReadWord

- **Prototype**

int ReadWord(unsigned int addr);

- **Description**

Read a data of word from the specified address .

- **Parameters**

addr[in] : the address to be read

The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 4 bytes.

For exemple： A word will be read from the address of 0x9a880 if addr[in]=0x9a880.

- **Include**

Peripheral_lib/Drvflash.h

- **Return Vaule**

The value of the word

- **Example**

```
/*  read the data from the address of 0x90880 */
ISP_FUNC_ROMP->FlashOpEn();
Int flag;  flag= ReadWord(0x90880);
ISP_FUNC_ROMP->FlashOpDis();
```

13.2.6. ReadPage

- **Prototype**

  int ReadPage(unsigned int addr, int* data);

- **Description**

  Read 32 data of word from the specified address in a row one time.

- **Parameters**

  addr[in] : the initial address to be read

  The input range is 0~0xffff, and the start address of flash is0x90000. The interval of address is 128(32*4) bytes, the page only could be read one by one, for only 128byte in each page . and the address only could be 0xuu00 or 0xuu80.

  For exemple：The address of 0x9a880 will be read if addr[in]=0x9a880.

  data [in] : storage the data to be read

  it could be 0~0xffffffff . The length of data[in] is 32 word

- **Include**

  Peripheral_lib/DrvFlash.h

- **Return Vaule**

  32 data of word

- **Example**

```
/* read 32 data of word from the address of 0x90880 in a row one time*/
unsigned int *A[32]={0};
ISP_FUNC_ROMP->FlashOpEn();
ReadPage(0x90880, A);
ISP_FUNC_ROMP->FlashOpDis();
```

13.2.7. ISP_FUNC_ROMP->SectorErase

- **Prototype**

  int ISP_FUNC_ROMP->SectorErase(uint32_t addr);

- **Description**

  Erase one sector to the specified address.

  Operating time about 2ms

- **Parameters**

  addr[in] : the initial address to Erase

  The input range is 0~0xffff, and the start address of flash is0x90000. Each sector include 32page.The interval of address is 128*32 bytes, and the first address is calculated by page,the address only could be 0xu000(u is defined by user)

  For exemple：The address of 0x91000 will Erase first if addr[in]=0x91000.

- **Include**

  Peripheral_lib/DrvFlash.h

- **Return Vaule**

© 2021-2022HYCON Technology Corp

0 : successful

3 : wrong address

- **Example**

/* Erase one sector from the inital address of 0x91000 */

ISP_FUNC_ROMP->FlashOpEn();

ISP_FUNC_ROMP->SectorErase(0x91000);

ISP_FUNC_ROMP->FlashOpDis();

13.2.8. ISP_FUNC_ROMP->CRC

- **Prototype**

Uint32_t ISP_FUNC_ROMP->CRC(uint32_t starta, uint32_t stopa)

- **Description**

CRC is calculated from starta to stopa+3.

Note: starta & stopa must be a multiple of 4, stopa-starta cannot be a multiple of 8. The CRC calculated in the entire FLASH space is 0x90000 to 0xAFFFC

- **Parameters**

starta [in] : The first address of the CRC to be calculated, a 20 bits address, the Flash space starts from 0x90000, the address value needs to be written with a 20 bits value, and the input range is 0x90000~0xAFFFC. If you want to start from 0x91000, the starta address parameter needs to fill in 0x91000.

stopa [in]: The last address of the CRC to be calculated, a 20 bits address. If the calculation ends at 0x913FF, the stopa address parameter needs to be filled with 0x913FC.

- **Include**

Peripheral_lib/DrvFlash.h

- **Return Vaule**

CRC value

- **Example**

/* calculate CRC from 0x91000 to 0x91BFF*/

ISP_FUNC_ROMP->FlashOpEn();

ISP_FUNC_ROMP->CRC(0x91000,0x91BFC);

ISP_FUNC_ROMP->FlashOpDis();

13.2.9. ISP_FUNC_ROMP->fastBlank

- **Prototype**

Uint32_t ISP_FUNC_ROMP->fastBlank(uint32_t starta, uint32_t stopa)

- **Description**

Fast Blank Check from starta to stopa+3.

Note: starta & stopa are multiples of 4

- **Parameters**

starta [in] : The first address of the CRC to be calculated, a 20 bits address, the Flash space starts from

0x90000, the address value needs to be written with a 20 bits value, and the input range is 0x90000~0xAFFFC. If you want to start from 0x91000, the starta address parameter needs to fill in 0x91000.

stopa [in]: The last address of the CRC to be calculated, a 20 bits address. If the calculation ends at 0x913FF, the stopa address parameter needs to be filled with 0x913FC.

- **Include**

  Peripheral_lib/DrvFlash.h

- **Return Vaule**

  0xFFFFFFFF: the area is blank

  Other: the area is not blank

- **Example**

```
/*Fast Blank Check from 0x91000 to 0x91BFF */
ISP_FUNC_ROMP->FlashOpEn();
ISP_FUNC_ROMP-> fastBlank (0x91000,0x91BFC);
ISP_FUNC_ROMP->FlashOpDis();
```

13.2.10. The storage structure of Flash

Sector & Page			
Sector	Page	Address Range	
0	0	0x90000	0x9007F
	1	0x90080	0x900FF
	...	...	..
	6	0x90300	0x9037F
	7	0x90380	0x903FF
1	8	0x90400	0x9047F
	9	0x90480	0x904FF
	...	...	...
	15	0x90780	0x907FF
...	...	...	...
127	1016	0xAFC00	0xAFC7f
	...	...	...
	1022	0xAFF00	0xAFF7F
	1023	0xAFF80	0xAFFFF

1 page= 32 word= 128 byte

1 sector= 8 page= 1K byte

© 2021-2022HYCON Technology Corp

14. Revision History

Version	Page	Revision Summary	The Date Of Revision
V05	ALL	First edition	2022/05/16
V06	All	1. Modify the interrupt HW9 setting of SYS_INTPriority() 2. Modify HAO frequency description 4M to 4.147M, 32M to 31.795M 3. Add DrvCLOCK_SelectIHOSC_CalHAO() function 4. Modify the WDT watchdog frequency division setting description 5. Modify the count trigger source parameters of DrvTMB1_Open and DrvTMB2_Open 6. Modify the parameters of DrvTMB_CPI1Input and DrvTMB2_CPI3Input 7. Remove DrvLCD_VLCDTrim 8. Corrected the note 2 of the Flash programming to change 1.8V to 2V	2022/12/12

© 2021-2022HYCON Technology Corp

15. C Library Change List

Date	Previous Version List		New version List	
2022/05/16	Version	Bug List	Version	Improvement
	V5.0	None		
2022/12/12	V5.0	1. HAO frequency correction 2. Modify WDT frequency divider content 3. DrvTMBx_Open parameter description error 4. TMB 2 _CPIxInput parameter description error	V6.0	1. Correct HAO frequency value 4M to 4.147M, 32M to 31.795M 2. Correct the content of WDT frequency division value 3. Modify 4.3.13, 4.3.30, 4.3.31, 4.3.42 input parameter description

© 2021-2022HYCON Technology Corp