



SPMC75X Family Programming Guide

V1.6 – Aug. 19, 2005

Important Notice

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

Table of Content

	<u>PAGE</u>
1 INTRODUCTION	1-1
1.1 INTRODUCTION	1-1
1.2 DEVICE STRUCTURE	1-1
1.3 THE CPU CORE	1-1
1.4 PERIPHERALS FUNCTION	1-1
1.5 SPECIAL FEATURES	1-2
1.6 FUNCTION BLOCK OVERVIEW	1-2
1.7 DEVICE FEATURE	1-4
1.8 SPMC75X FAMILY MCU PACKAGES	1-6
1.8.1 SPMC75F2413A-PQ041(QFP64 Package, Pitch 1.0 mm).....	1-6
1.8.2 SPMC75F2413A-PQ051(QFP80 Package, Pitch 0.8 mm).....	1-7
1.8.3 SPMC75F2313A-PD161(SDIP42 Package, Pitch 1.78 mm).....	1-8
1.8.4 SPMC75F2313A-PL011(LQFP44 Package, Pitch 0.8 mm).....	1-9
1.9 SPMC75X FAMILY MCU PIN DESCRIPTIONS	1-10
1.9.1 SPMC75F2413A 80-Pin QFP signals list.....	1-10
1.9.2 SPMC75F2413A 64-Pin QFP signals list.....	1-12
1.9.3 SPMC75F2313A 42-Pin SDIP signals list.....	1-14
1.9.4 SPMC75F2313A 44-Pin LQFP signals list.....	1-15
1.10 DEVELOPMENT SUPPORT	1-18
2 OSCILLATOR	2-1
2.1 OSCILLATOR	2-1
2.1.1 200KHz R-Oscillator.....	2-1
2.1.2 Phase-lock Loop	2-1
2.1.3 Clock Monitoring	2-2
2.2 FUNCTION BLOCK	2-2
2.3 SYSTEM CLOCK CONTROL REGISTER.....	2-3
2.3.1 P_Clk_Ctrl (0x7007): System Clock Control Register	2-3
2.4 APPLICATION CIRCUIT	2-3
2.5 DESIGN TIPS.....	2-4
3 RESET	3-1
3.1 INTRODUCTION	3-1
3.2 POWER-UP PROCEDURE	3-1
3.3 RESET MODE.....	3-2

3.3.1 Power on reset (POR).....	3-2
3.3.2 External Reset.....	3-2
3.3.3 Low Voltage Reset (LVR).....	3-3
3.3.4 Watch Dog Timer Reset (WDTR).....	3-3
3.3.5 Illegal address reset (IAR).....	3-4
3.3.6 Illegal instruction reset (IIR)	3-4
3.4 RESET SOURCE TABLE.....	3-5
3.5 CONTROL REGISTER	3-5
3.5.1 P_Reset_Status(0x7006): Reset Status Register	3-5
3.6 DESIGN TIPS.....	3-7
3.6.1 Reset Status.....	3-7
4 MCU	4-1
4.1 FUNCTION BLOCK	4-2
4.2 CPU INSTRUCTION AND EXECUTIONS CYCLE	4-3
5 MEMORY ORGANIZATION	5-1
5.1 INTRODUCTION	5-1
5.2 FLASH	5-2
5.3 SRAM	5-2
5.4 I/O PORT REGISTER	5-3
5.5 RESET AND INTERRUPT VECTOR	5-6
6 FLASH ORGANIZATION AND CONTROL	6-1
6.1 INTRODUCTION	6-1
6.2 FLASH OPERATION	6-1
6.3 FLASH REGISTERS ADDRESS TABLE.....	6-3
6.4 CONTROL REGISTERS	6-3
6.4.1 P_Flash_RW (0x704D): Embedded Flash Access Control Register	6-3
6.4.2 P_Flash_Cmd (0x7555): Embedded flash Access Command Register	6-4
6.4.3 P_System_Option (0x8000): System Option Register.....	6-4
6.5 FLASH SECURITY PROTECTION	6-5
7 INTERRUPT	7-1
7.1 INTRODUCTION	7-1
7.2 INTERRUPT PROCEDURE	7-1
7.3 INTERRUPT SOURCES.....	7-3
7.4 INTERRUPT REGISTERS ADDRESS TABLE.....	7-6
7.5 CONTROL REGISTERS	7-6

7.5.1 P_INT_Status (0x70A0): Interrupt Status Register	7-6
7.5.2 P_INT_Priority (0x70A4): IRQ and FIQ Priority Selection Register.....	7-8
7.5.3 P_MisINT_Ctrl (0x70A8): Miscellaneous Interrupt Control Register	7-10
7.6 INFORMATION SAVING DURING INTERRUPTS	7-10
7.7 INTERRUPTS PROTOTYPE.....	7-11
7.8 DESIGN TIPS.....	7-16
8 I/O PORTS	8-1
8.1 INTRODUCTION	8-1
8.1.1 Large Driving Pins.....	8-4
8.1.2 Key-change Interrupt Pins	8-4
8.2 IO PORT REGISTERS ADDRESS TABLE	8-5
8.3 PORT A	8-6
8.3.1 P_IOA_Data (0x7060) : IO Port A Data Register.....	8-6
8.3.2 P_IOA_Buffer (0x7061) : IO Port A Buffer Register.....	8-6
8.3.3 P_IOA_Dir (0x7062) : IO Port A Direction Register	8-7
8.3.4 P_IOA_Attrib (0x7063) : IO Port A Attribute Register	8-7
8.3.5 P_IOA_Latch (0x7064) : IO Port A Latch Register	8-7
8.3.6 P_IOA_SPE (0x7080) : IO Port A Special Function Enable Register.....	8-8
8.3.7 P_IOA_KCER (0x7084) : IO Port A Key Change Enable Register	8-9
8.3.8 Special Functions of Port A	8-10
8.4 PORT B	8-11
8.4.1 P_IOB_Data (0x7068) : IO Port B Data Register	8-11
8.4.2 P_IOB_Buffer (0x7069) : IO Port B Buffer Register	8-11
8.4.3 P_IOB_Dir (0x706A) : IO Port B Direction Register	8-12
8.4.4 P_IOB_Attrib (0x706B) : IO Port B Attribute Register	8-12
8.4.5 P_IOB_SPE (0x7081) : IO Port B Special Function Enable Register	8-12
8.4.6 Special Function of Port B.....	8-14
8.5 PORT C	8-15
8.5.1 P_IOC_Data (0x7070) : IO Port C Data Register.....	8-15
8.5.2 P_IOC_Buffer (0x7071) : IO Port C Buffer Register.....	8-15
8.5.3 P_IOC_Dir (0x7072) : IO Port C Direction Register	8-16
8.5.4 P_IOC_Attrib (0x7073) : IO Port C Attribute Register	8-16
8.5.5 P_IOC_SPE (0x7082) : IO Port C Special Function Enable Register.....	8-16
8.5.6 Special Function of Port C	8-18
8.6 PORT D	8-19
8.6.1 P_IOD_Data (0x7078) : IO Port D Data Register.....	8-19
8.6.2 P_IOD_Buffer (0x7079) : IO Port D Buffer Register.....	8-19

8.6.3 P_IOD_Dir (0x707A) : IO Port D Direction Register.....	8-20
8.6.4 P_IOD_Attrib (0x707B) : IO Port D Attribute Register.....	8-20
8.6.5 Special Function of Port D	8-20
8.7 I/O PROGRAMMING CONSIDERATIONS.....	8-21
8.8 I/O INITIALIZATION.....	8-21
9 POWER SAVING MODES AND WAKEUP	9-1
9.1 INTRODUCTION	9-1
9.2 WAKE-UP SOURCES	9-3
9.3 CONTROL REGISTER	9-3
9.3.1 P_Wait_Enter (0x700C) : Wait-mode Entrance Register	9-3
9.3.2 P_Standby_Enter (0x700E) : Standby-mode Entrance Register.....	9-4
9.3.3 P_Wakeup_Ctrl (0x700F) : Wake-up Control Register	9-4
9.4 DESIGN TIPS.....	9-5
10 PDC TIMER 0 AND 1 MODULE	10-1
10.1 INTRODUCTION	10-1
10.2 PDC TIMERS FEATURES	10-1
10.3 PDC TIMERS INPUT/OUTPUT/SPECIAL FUNCTION PINS	10-4
10.4 PDC TIMER COUNTING OPERATION.....	10-5
10.4.1 Continuous Up Counting Mode with Edge-Aligned PWM.....	10-5
10.4.2 Continuous up/down counting mode with Center-Aligned PWM	10-7
10.5 REGISTERS DESCRIPTIONS.....	10-10
10.5.1 Common.....	10-10
10.5.2 PDC Timer 0.....	10-10
10.5.3 PDC Timer 1.....	10-10
10.6 TIMER 0 AND 1 CONTROL REGISTERS	10-11
10.6.1 P_TMR0_Ctrl (0x7400): Timer 0 Control Register.....	10-11
10.6.2 P_TMR1_Ctrl (0x7401): Timer 1 Control Register.....	10-11
10.7 TIMER 0 AND 1 INPUT AND OUTPUT CONTROL REGISTER	10-12
10.7.1 P_TMR0_IOCctrl (0x7410) : Timer 0 IO control register	10-13
10.7.2 P_TMR1_IOCctrl (0x7411) : Timer 1 IO control register.....	10-13
10.8 TIMER 0 AND 1 INTERRUPT ENABLE REGISTER	10-13
10.8.1 P_TMR0_INT (0x7420): Timer 0 Interrupt Enable Register.....	10-14
10.8.2 P_TMR1_INT (0x7421): Timer 1 Interrupt Enable Register.....	10-14
10.9 TIMER 0 AND 1 INTERRUPT STATUS REGISTER.....	10-15
10.9.1 P_TMR0_Status (0x7425): Timer 0 Interrupt Status Register.....	10-15
10.9.2 P_TMR1_Status (0x7426): Timer 1 Interrupt Status Register.....	10-15
10.10 TIMER START REGISTER	10-16

10.10.1 P_TMR_Start (0x7405): Timer Counter Start Register	10-17
10.11 TIMER 0 AND 1 POSITION DETECTION CONTROL REGISTER	10-17
10.11.1 P_POS0_DectCtrl (0x7462): Timer 0 Position Detection Control Register.....	10-18
10.11.2 P_POS1_DectCtrl (0x7463): Timer 1 Position Detection Control Register.....	10-18
10.12 TIMER 0 AND 1 POSITION DETECTION DATA REGISTER	10-19
10.12.1 P_POS0_DectData (0x7464): Timer 0 Position Detection Data Register	10-19
10.12.2 P_POS1_DectData (0x7465): Timer 1 Position Detection Data Register	10-19
10.13 TIMER 0 AND 1 COUNTER REGISTER	10-19
10.13.1 P_TMR0_TCNT (0x7430): Timer 0 Counter Register	10-20
10.13.2 P_TMR1_TCNT (0x7431): Timer 1 Counter Register	10-20
10.14 TIMER 0 AND 1 GENERAL AND BUFFER REGISTER	10-20
10.14.1 P_TMR0_TGRA (0x7440): Timer 0 General Register A.....	10-21
10.14.2 P_TMR0_TGRB (0x7441): Timer 0 General Register B	10-21
10.14.3 P_TMR0_TGRC (0x7442): Timer 0 General Register C	10-21
10.14.4 P_TMR1_TGRA (0x7443): Timer 1 General Register A.....	10-21
10.14.5 P_TMR1_TGRB (0x7444): Timer 1 General Register B	10-21
10.14.6 P_TMR1_TGRC (0x7445): Timer 1 General Register C	10-21
10.14.7 P_TMR0_TBRA (0x7450): Timer 0 Buffer Register A.....	10-21
10.14.8 P_TMR0_TBRB (0x7451): Timer 0 Buffer Register B	10-21
10.14.9 P_TMR0_TBRC (0x7452): Timer 0 Buffer Register C	10-21
10.14.10 P_TMR1_TBRA (0x7453): Timer 1 Buffer Register A.....	10-21
10.14.11 P_TMR1_TBRB (0x7454): Timer 1 Buffer Register B	10-21
10.14.12 P_TMR1_TBRC (0x7455): Timer 1 Buffer Register C	10-21
10.15 TIMER 0 AND 1 PERIOD REGISTER	10-22
10.15.1 P_TMR0_TPR (0x7435): Timer 0 Period Register	10-22
10.15.2 P_TMR1_TPR (0x7436): Timer 1 Period Register	10-22
10.16 PDC TIMERS OPERATION	10-22
10.16.1 Normal Counting Operation	10-22
10.16.2 PWM Compare Match Output Operation.....	10-24
10.16.3 Input Capture Operation	10-25
10.16.4 Position Detection Change (PDC) Mode Operation	10-28
10.16.5 Phase Counting Mode Operation	10-30
10.16.6 Phase Counting Mode 1	10-30
10.16.7 Phase Counting Mode 2	10-31
10.16.8 Phase Counting Mode 3	10-32
10.16.9 Phase Counting Mode 4	10-33
10.17 APPLICATION EXAMPLE DESIGN TIPS	10-34

11 TPM TIMER 2 MODULE.....11-1

11.1 INTRODUCTION	11-1
11.2 TIMER 2 FEATURES	11-1
11.3 TPM TIMER 2 INPUT/OUTPUT/SPECIAL FUNCTION PINS.....	11-3
11.4 TPM TIMER 2 OPERATION.....	11-3
11.4.1 Continuous Up Counting Mode with Edge-Aligned PWM.....	11-4
11.4.2 Continuous up/down counting mode with Center-Aligned PWM	11-6
11.5 REGISTERS DESCRIPTIONS.....	11-7
11.5.1 Common.....	11-7
11.5.2 TPM Timer 2.....	11-7
11.6 TIMER 2 CONTROL REGISTERS	11-8
11.6.1 P_TMR2_Ctrl (0x7402): Timer 2 Control Register.....	11-8
11.7 TIMER 2 INPUT AND OUTPUT CONTROL REGISTER	11-9
11.7.1 P_TMR2_IOCtrl (0x7412) : Timer 2 IO control register	11-9
11.8 TIMER 2 INTERRUPT ENABLE REGISTER	11-10
11.8.1 P_TMR2_INT (0x7422): Timer 2 Interrupt Enable Register.....	11-10
11.9 TIMER 2 INTERRUPT STATUS REGISTER	11-11
11.9.1 P_TMR2_Status (0x7427): Timer 2 Interrupt Status Register.....	11-11
11.10 TIMER START REGISTER	11-12
11.10.1 P_TMR_Start (0x7405): Timer Counter Start Register.....	11-12
11.11 TIMER 2 COUNTER REGISTER	11-13
11.11.1 P_TMR2_TCNT (0x7432): Timer 2 Counter Register.....	11-13
11.12 TIMER 2 GENERAL AND BUFFER REGISTER	11-13
11.12.1 P_TMR2_TGRA (0x7446): Timer 2 General Register A	11-14
11.12.2 P_TMR2_TGRB (0x7447): Timer 2 General Register B	11-14
11.12.3 P_TMR2_TBRA (0x7456): Timer 2 Buffer Register A	11-14
11.12.4 P_TMR2_TBRB (0x7457): Timer 2 Buffer Register B.....	11-14
11.13 TIMER 2 PERIOD REGISTER	11-14
11.13.1 P_TMR2_TPR (0x7437): Timer 2 Period Register.....	11-15
11.14 TPM TIMER 2 OPERATION	11-15
11.14.1 Normal Counting Operation	11-15
11.14.2 PWM Compare Match Output Operation	11-16
11.14.3 Input Capture Operation.....	11-17
11.15 APPLICATION EXAMPLE DESIGN TIPS	11-19
12 MCP TIMER 3 AND 4 MODULE	12-1
12.1 INTRODUCTION	12-1
12.2 MCP TIMER 3 AND 4 FEATURES	12-1
12.3 MCP TIMER 3 AND 4 INPUT/OUTPUT/SPECIAL FUNCTION PINS.....	12-4

12.4 MCP TIMER 3 AND 4 COUNTING OPERATION.....	12-5
12.4.1 Continuous Up Counting Mode with Edge-Aligned PWM.....	12-5
12.4.2 Continuous up/down counting mode with Center-Aligned PWM	12-7
12.5 REGISTERS DESCRIPTIONS.....	12-8
12.5.1 Common.....	12-8
12.5.2 MCP Timer 3	12-8
12.5.3 MCP Timer 4	12-8
12.6 TIMER 3 AND 4 CONTROL REGISTERS	12-9
12.6.1 P_TMR3_Ctrl (0x7403) : Timer 3 Control Register.....	12-9
12.6.2 P_TMR4_Ctrl (0x7404) : Timer 4 Control Register.....	12-9
12.7 TIMER 3 AND 4 INPUT AND OUTPUT CONTROL REGISTER	12-10
12.7.1 P_TMR3_IOCtrl (0x7413) : Timer 3 IO control register	12-11
12.7.2 P_TMR4_IOCtrl (0x7414) : Timer 4 IO control register	12-11
12.8 TIMER 3 AND 4 INTERRUPT ENABLE REGISTER	12-11
12.8.1 MCP timer 3 channel interrupt sources.....	12-11
12.8.2 MCP timer 4 channel interrupt sources.....	12-12
12.8.3 P_TMR3_INT (0x7423): Timer 3 Interrupt Enable Register.....	12-12
12.8.4 P_TMR4_INT (0x7424): Timer 4 Interrupt Enable Register.....	12-12
12.9 TIMER 3 AND 4 INTERRUPT STATUS REGISTER.....	12-13
12.9.1 P_TMR3_Status (0x7428): Timer 3 Interrupt Status Register.....	12-13
12.9.2 P_TMR4_Status (0x7429): Timer 4 Interrupt Status Register.....	12-13
12.10 TIMER OUTPUT ENABLE REGISTER.....	12-14
12.10.1 P_TMR_Output (0x7406): Timer Output Enable Register.....	12-14
12.11 TIMER 3 AND 4 OUTPUT CONTROL REGISTER	12-16
12.11.1 P_TMR3_OutputCtrl (0x7407): Timer 3 Output Control Register	12-17
12.11.2 P_TMR4_OutputCtrl (0x7408): Timer 4 Output Control Register	12-17
12.12 TIMER 3 AND 4 DEAD TIME AND CONTROL REGISTER	12-20
12.12.1 P_TMR3_DeadTime (0x7460): Timer 3 Dead Time and Control Register.....	12-21
12.12.2 P_TMR4_DeadTime (0x7461): Timer 4 Dead Time and Control Register.....	12-21
12.13 TIMER FAULT INPUT CONTROL REGISTER.....	12-22
12.13.1 P_Fault1_Ctrl (0x7466): Fault input 1 Control and Status Register	12-25
12.13.2 P_Fault2_Ctrl (0x7467): Fault input 2 Control and Status Register	12-25
12.14 TIMER FAULT RELEASE REGISTER	12-26
12.14.1 P_Fault1_Release(0x746A): Fault 1 Flag Release Register.....	12-26
12.14.2 P_Fault2_Release(0x746B): Fault 2 Flag Release Register.....	12-26
12.15 TIMER OVERLOAD PROTECTION CONTROL AND STATUS REGISTER	12-27
12.15.1 P_Ol1_Ctrl(0x7468): Overload Input 1 Control and Status Register.....	12-30
12.15.2 P_Ol2_Ctrl(0x7469): Overload input 2 Control and Status Register.....	12-30

12.16 TIMER/PWM MODULE WRITE ENABLE CONTROL REGISTER	12-31
12.16.1 P_TPWM_Write (0x7409): Timer/PWM Module Write Enable Control Register.....	12-32
12.17 TIMER LOAD-OK REGISTER	12-32
12.17.1 P_TMR_LDOK (0x740A) : Timer Load-OK Register	12-32
12.18 TIMER START REGISTER	12-33
12.18.1 P_TMR_Start (0x7405): Timer Counter Start Register	12-33
12.19 TIMER 3 AND 4 COUNTER REGISTER	12-34
12.19.1 P_TMR3_TCNT (0x7433): Timer 3 Counter Register	12-34
12.19.2 P_TMR4_TCNT (0x7434): Timer 4 Counter Register	12-34
12.20 TIMER 3 AND 4 GENERAL AND BUFFER REGISTER	12-34
12.20.1 P_TMR3_TGRA (0x7448): Timer 3 General Register A.....	12-35
12.20.2 P_TMR3_TGRB (0x7449): Timer 3 General Register B	12-35
12.20.3 P_TMR3_TGRC (0x744A): Timer 3 General Register C.....	12-35
12.20.4 P_TMR3_TGRD (0x744B): Timer 3 General Register D.....	12-35
12.20.5 P_TMR4_TGRA (0x744C): Timer 4 General Register A	12-35
12.20.6 P_TMR4_TGRB (0x744D): Timer 4 General Register B	12-35
12.20.7 P_TMR4_TGRC (0x744E): Timer 4 General Register C.....	12-35
12.20.8 P_TMR4_TGRD (0x744F): Timer4 General Register D.....	12-35
12.20.9 P_TMR3_TBRA (0x7458): Timer 3 Buffer Register A.....	12-35
12.20.10 P_TMR3_TBRB (0x7459): Timer 3 Buffer Register B	12-35
12.20.11 P_TMR3_TBRC (0x745A): Timer 3 Buffer Register C.....	12-35
12.20.12 P_TMR4_TBRA (0x745C): Timer 4 Buffer Register A	12-35
12.20.13 P_TMR4_TBRB (0x745D): Timer 4 Buffer Register B	12-35
12.20.14 P_TMR4_TBRC (0x745E): Timer 4 Buffer Register C.....	12-35
12.21 TIMER 3 AND 4 PERIOD REGISTER	12-36
12.21.1 P_TMR3_TPR (0x7438): Timer 3 Period Register	12-36
12.21.2 P_TMR4_TPR (0x7439): Timer 4 Period Register	12-36
12.22 MCP TIMER 3 AND 4 OPERATION	12-36
12.22.1 Normal Counting Operation	12-36
12.22.2 PWM Output Operation	12-37
12.23 APPLICATION EXAMPLE DESIGN TIPS	12-38
13 COMPARE MATCH TIMER	13-1
13.1 INTRODUCTION	13-1
13.2 COMPARE MATCH TIMER REGISTERS ADDRESS TABLE	13-2
13.3 CONTROL REGISTERS	13-2
13.3.1 P_CMT_Start (0x7500) : Compare Match Timer Start Register	13-2

13.3.2 P_CMT_Ctrl (0x7501) : Compare Match Timer Control and Status Register	13-2
13.3.3 P_CMT0_TCNT (0x7508) : Compare Match Timer 0 Counter Register	13-4
13.3.4 P_CMT1_TCNT (0x7509) : Compare Match Timer 1 Counter Register	13-4
13.3.5 P_CMT0_TPR (0x7510) : Compare Match Timer 0 Period Register	13-4
13.3.6 P_CMT1_TPR (0x7511) : Compare Match Timer 1 Period Register	13-4
13.4 DESIGN TIPS.....	13-4
14 TIME BASE MODULE AND BUZZER UNIT	14-1
14.1 INTRODUCTION	14-1
14.2 TIMER BASE MODULE AND BUZZER REGISTERS ADDRESS TABLES.....	14-1
14.3 CONTROL REGISTERS	14-1
14.3.1 P_TMB_Reset (0x70B8) : Time Base Reset Register	14-1
14.3.2 P_BZO_Ctrl (0x70B9) : Buzzer Output Control Register	14-2
14.4 DESIGN TIPS.....	14-2
15 STANDARD PERIPHERAL INTERFACE, SPI	15-1
15.1 INTRODUCTION	15-1
15.2 SPI CONTROL PINS CONFIGURATION	15-2
15.3 SPI REGISTERS ADDRESS TABLE.....	15-2
15.4 CONTROL REGISTER	15-3
15.4.1 P_SPI_Ctrl (0x7140): SPI Control Register	15-3
15.4.2 P_SPI_TxStatus (0x7141): SPI Transmit Status Register	15-4
15.4.3 P_SPI_TxBuf (0x7142): SPI Transmission Buffer Register	15-4
15.4.4 P_SPI_RxStatus 0x7143): SPI Receive Status Register.....	15-5
15.4.5 P_SPI_RxBuf (0x7144): SPI Reception Buffer Register	15-5
15.5 SPI OPERATION.....	15-6
15.5.1 Master Mode	15-6
15.5.2 Slave Mode	15-7
15.6 DESIGN TIPS.....	15-8
16 UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER – UART	16-1
16.1 INTRODUCTION	16-1
16.2 APPLICATION CIRCUIT	16-2
16.2.1 UART Frame Scheme.....	16-2
16.3 UART CONTROL PIN CONFIGURATION	16-2
16.4 UART REGISTERS ADDRESS TABLE.....	16-3
16.5 CONTROL REGISTER	16-3
16.5.1 P_UART_Data (0x7100): UART Data Register	16-3
16.5.2 P_UART_RXStatus (0x7101): UART Reception Error Flag Register	16-4

16.5.3 P_UART_Ctrl (0x7102): UART Control Register	16-4
16.5.4 P_UART_BaudRate(0x7103): UART Baud Rate Setup Register	16-6
16.5.5 P_UART_Status (0x7104): UART Status Register.....	16-6
16.6 UART OPERATION.....	16-7
16.7 DESIGN TIPS.....	16-10
17 10-BIT A/D CONVERTER.....	17-1
17.1 INTRODUCTION	17-1
17.2 ADC REGISTERS ADDRESS TABLE.....	17-2
17.3 CONTROL REGISTERS	17-2
17.3.1 P_ADC_Setup (0x7160) : ADC Setup Register	17-3
17.3.2 P_ADC_Ctrl(0x7161) : ADC Control Register	17-4
17.3.3 P_ADC_Channel(0x7166H) : ADC Input Channels Select Register.....	17-5
17.3.4 P_ADC_Data (0x7162H) : ADC Data Register.....	17-6
17.4 ADC OPERATION.....	17-6
17.5 DESIGN TIPS.....	17-7
18 WATCHDOG	18-1
18.1 INTRODUCTION	18-1
18.2 WDT REGISTERS ADDRESS TABLE	18-2
18.3 CONTROL REGISTERS	18-2
18.3.1 P_WatchDog_Ctrl (0x700A) : Watchdog Control Register.....	18-2
18.3.2 P_WatchDog_Clr (0x700B) : Watchdog Clear Register	18-3
18.4 DESIGN TIPS.....	18-3
19 INSTRUCTION SET.....	19-1
19.1 INTRODUCTION	19-1
19.2 ALU OPERATION	19-2
19.3 ADDRESSING MODES	19-3
19.4 REGISTER.....	19-4
19.5 BRANCH CONDITION	19-5
19.6 SHIFT	19-6
19.7 INSTRUCTION SET	19-7
20 HARDWARE SUMMARY.....	20-2
20.1 PROGRAMMABLE REGISTERS OF CPU ON SPMC75X FAMILY MCU.....	20-2
20.2 PROGRAMMABLE REGISTERS OF WATCHDOG UNIT ON SPMC75X FAMILY MCU	20-2
20.3 PROGRAMMABLE REGISTERS OF POWER-SAVING UNIT ON SPMC75X FAMILY MCU.....	20-3
20.4 PROGRAMMABLE REGISTERS OF I/O PORT ON SPMC75X FAMILY MCU	20-4

20.5 PROGRAMMABLE REGISTERS OF INTERRUPT UNIT ON SPMC75X FAMILY MCU	20-11
20.6 PROGRAMMABLE REGISTERS OF TIMERS UNIT ON SPMC75X FAMILY MCU	20-12
20.7 PROGRAMMABLE REGISTERS OF CMT TIMER ON SPMC75X FAMILY MCU	20-22
20.8 PROGRAMMABLE REGISTERS OF TMB / BUZZER UNIT ON SPMC75X FAMILY MCU.....	20-23
20.9 PROGRAMMABLE REGISTERS OF SPI ON SPMC75X FAMILY MCU	20-24
20.10 PROGRAMMABLE REGISTERS OF UART ON SPMC75X FAMILY MCU	20-25
20.11 PROGRAMMABLE REGISTERS OF ADC ON SPMC75X FAMILY MCU	20-27
20.12 PROGRAMMABLE REGISTERS OF WATCHDOG TIMER ON SPMC75X FAMILY MCU	20-28
21 SPMC75X FAMILY MCU DEVELOPMENT SYSTEM	21-1
21.1 DEVELOPMENT SYSTEM SETUP DIAGRAM.....	21-1
21.2 SPMC75F2413A EVM BOARD v1.1	21-1
21.3 SPMC75F2413A EVM BOARD v1.1 SCHEMATICS	21-2
22 SUNPLUS U'NSP IDE DEVELOPMENT ENVIRONMENT	22-1
22.1 GENERAL DESCRIPTION	22-1
22.2 INSTALLATION	22-1
22.3 QUICK START.....	22-3
22.4 GET TO KNOW THE µ'NSP® IDE	22-3
22.4.1 Main Window.....	22-3
22.4.2 Workspace Window	22-4
22.4.3 Output Window.....	22-4
22.4.4 Edit Window	22-5
22.4.5 Binary Editor.....	22-5
22.4.6 Debug Window.....	22-6
22.4.7 Register Window	22-7
22.4.8 Command Window.....	22-9
22.4.9 Breakpoint Window	22-11
22.4.10 Watch Window	22-12
22.5 PROJECT	22-15
22.5.1 Project Window	22-15
22.5.2 Create Project	22-16
22.5.3 Add File and Resource to Project	22-17
22.5.4 Save Project.....	22-18
22.5.5 Create Object.....	22-20
22.5.6 Use Group in Project.....	22-20
22.5.7 Add Group to Project.....	22-21
22.5.8 Change Group Property	22-21
22.5.9 Move File and Group.....	22-22

22.5.10 Select Body.....	22-22
22.5.11 Set Option for Project.....	22-23
22.5.12 Build Project.....	22-31
22.5.13 Run Program	22-32
22.5.14 Load Program	22-32
22.6 SOURCE CODE	22-32
22.6.1 Create Document	22-32
22.6.2 Save Document.....	22-33
22.6.3 Bookmark	22-33
22.6.4 Find Text.....	22-34
22.6.5 Binary Editor.....	22-35
22.7 RESOURCE	22-35
22.7.1 Resource ID	22-35
22.7.2 Change Resource	22-36
22.7.3 SUNPLUS format.....	22-36
22.8 DEBUGGER.....	22-37
22.8.1 Run Control	22-37
22.8.2 Debug window.....	22-38
22.9 PROFILE	22-39
22.9.1 Profile Function	22-39
22.9.2 Profile Operation	22-39
22.10 TOOL	22-41
22.10.1 Customize Developer Studio	22-41
22.10.2 Tool	22-41
22.10.3 Keyboard.....	22-43
22.10.4 Editor.....	22-44
22.10.5 Format.....	22-45
22.10.6 Editor.....	22-45
22.10.7 External Editor	22-46
22.10.8 Directories.....	22-46
22.10.9 Debug	22-47
22.10.10 Body Display	22-48
22.11 OTHER TOOL.....	22-48
22.11.1 Dump File	22-48
22.11.2 LibMaker.....	22-51
22.11.3 MemoryMap	22-52
22.12 HOT KEY.....	22-53
22.12.1 Toolbar Button.....	22-53

23 SUNPLUS DMC TOOLKIT INTRODUCTION	23-1
23.1 DMC TOOLKIT FEATURES	23-1
23.1.1 Controlling System	23-1
23.1.2 Monitoring System	23-1
23.2 INSTALLATION	23-1
23.2.1 System Requirements.....	23-1
23.3 LOOKING THROUGH DMC TOOLKIT COMPONENTS.....	23-3
23.3.1 Monitoring Window.....	23-4
23.3.2 Charting Window.....	23-6
23.4 QUICK START.....	23-7
23.4.1 Launching the DMC Toolkit	23-7
23.4.2 Controlling and Monitoring System	23-8
23.4.3 Starting or Stopping Motor	23-10
23.4.4 Saving Project	23-10
23.4.5 Exiting DMC Toolkit.....	23-10
23.5 APPLYING DMC TOOLKIT	23-10
23.5.1 Opening an Existing File	23-10
23.5.2 Setting for Variable Update Rate.....	23-10
23.5.3 Starting Time Function	23-11
23.5.4 Editing Charting Window.....	23-11
23.6 ACCELERATOR	23-13
23.7 TECHNICAL SUPPORT	23-13
24 APPENDIX A. RESERVED WORDS	24-1

Document Revision History

Revision	Date	By	Remark
1.6	08/19/2005	Max Huang	1. ADC conversion clock rate is limited to 1.5 MHz. 2. Modified SPIFS description in P_SPI_Ctrl register. 3. Rename IOxMOD to IOxMODE. 4. Modified description of P_Fault1/2_Ctrl register. 5. Replace with new version EVM board circuit.
1.5	12/30/2004	Max Huang	1. Corrects the figure 11-10. 2. Corrects the P_Fault1_Ctrl and P_Fault2_Ctrl registers. 3. Add Table 13-5 and 13-6. 4. Add description of P_ADC_Setup register. 5. Add description of clock pins : XTAL1, XTAL2. 6. Add description of P_Olx_Ctrl register. 7. Add clock application circuit in section 3.4. 8. Add input capture configuration settings in section 11 and 12.
1.4	10/12/2004	Max Huang	1. in page 16, CPU maximum operating speed 24.0 MHz 2. in page 17, remove halt mode. 3. Modified section 2.9 pin description. 4. Modified section 2.1 document description. 5. Fix the bit 8 of P_Reset_Status register. 6. Modified section 5 MCU architecture descriptions. 7. Add addressing modes descriptions in section 20. 8. Add IRQ5 in section 8.1 description and modified table 8-1. 9. Update GPIO Specification :Key change of PA[15:8] interrupt. 10. Rename SPMC75xxxx to SPMC75F2413A. 11. Rename TPM0/1 to PDC0/1. 12. Rename TPM3/4 to MCP3/4. 13. Modify figure 18-1and 18-2. 14. Modify figure 16-2.
1.3	08/13/2004	Max Huang	1. Add and modified motor control timers descriptions. 2. Remove P_IOC_Latch description in table 9-2. 3. Add EVM board circuit of SPMC75F2413A.
1.2	08/03/2004	Max Huang	1. Remove P_IOB_Latch, P_IOC_Latch, P_IOD_Latch registers in table 6-2. 2. Remove P_IOB_Latch, P_IOC_Latch registers in table 9-2. 3. Correct the initial value of P_Iox_Attrib (x = A, B, C, D) registers. 4. Correct the attribute of P_Wait_Enter and P_Standby_Enter registers. 5. Correct the initial value of P_TMRx_TPR (x = 0, 1, 2, 3, 4) registers. 6. Correct the attribute of P_BZO_Ctrl register. 7. Correct the TXCHSEL bit attribute of P_UART_Ctrl register. 8. Remove the VRXEN bit of P_ADC_Setup register.
1.1	06/24/2004	Lanin Lin and Max Huang	Add P_TMR_LDOK port & Add UART TX & RX enable bit
1.0	04/20/2004	Lanin Lin and Max Huang	Preliminary Release Version

1 Introduction

1.1 Introduction

The SPMC75X family MCU are the embedded 16-bit micro-controller, designed for inverter-fed motor driver, power electronics, home application and car fan control system, etc. This document explains the SPMC75X family MCU architecture and each peripheral modules.

1.2 Device Structure

Each part of a device can be defined into three groups:

1. SUNPLUS 16-bit u'nSP processor (ISA 1.2) CPU core
2. Peripherals Function.
3. Special Function Features

1.3 The CPU Core

The core pertains to the basic features that are the requirements for making the device to operate. These include:

1. Device Oscillator
2. Reset logic
3. CPU (Central Processing Unit) operation
4. ALU (Arithmetic Logical Unit) operation
5. Device memory map organization
6. Interrupt operation
7. Instruction set

1.4 Peripherals Function

Peripherals functions are the features that add a differentiation from a microprocessor. These functions facilitate interfacing to the external world (such as general purpose I/O, A/D inputs, and PWM outputs), and internal tasks such as keeping different time bases (such as timers). The peripherals that are discussed are:

1. General purpose I/O
2. Timer
3. Capture and PWM compare match output
4. Standard Peripheral Interface (SPI)
5. UART
6. CompareMatch Timer (CMT)
7. Time Base and Buzzer Output
8. 10-bit Analog to Digital Converter (ADC)

1.5 Special Features

Special features are the unique features to help performing one or more of the following tasks:

1. Lower system cost
2. Increase system reliability
3. Increase design flexibility

The SPMC75X family MCU offer several features to help achieving these goals.

1. On-chip Power-on Reset (POR)
2. Low Voltage Reset (LVR)
3. Watchdog Timer
4. Low power mode (Wait/Standby)
5. In-Circuit Serial Programming
6. Suitable for air conditioner outdoor/indoor unit
7. Can direct drive (output 3-phase 6-pin PWM waveforms) two BLDC (Brushless DC) or AC induction motors simultaneously.

1.6 Function Block Overview

The SPMC75X family MCU devices are specially designed for motor control applications. It is equipped with 16-bit u'nSP CPU, embedded Flash, SRAM, and other functional modules. The included functional modules are

- Clock Generation Module (3 ~ 6MHz crystal input, 12 ~ 24 MHz CPU clock speed with PLL module)
- General Purpose IO Port
- Interrupt Control Module
- Reset Management Module
- Two channels of PDC Timers, PDC Timer 0 and Timer 1.
- TPM Timer 2 Module.
- Two channel of MCP Timers, MCP Timer 3 and Timer 4.
- Compare Match Timer
- Time Base Module and Buzzer unit
- A/D converter : 10 bits, 8 channel, Analog-to-Digital Converter
- Serial Communication Interface
- Watchdog Timer

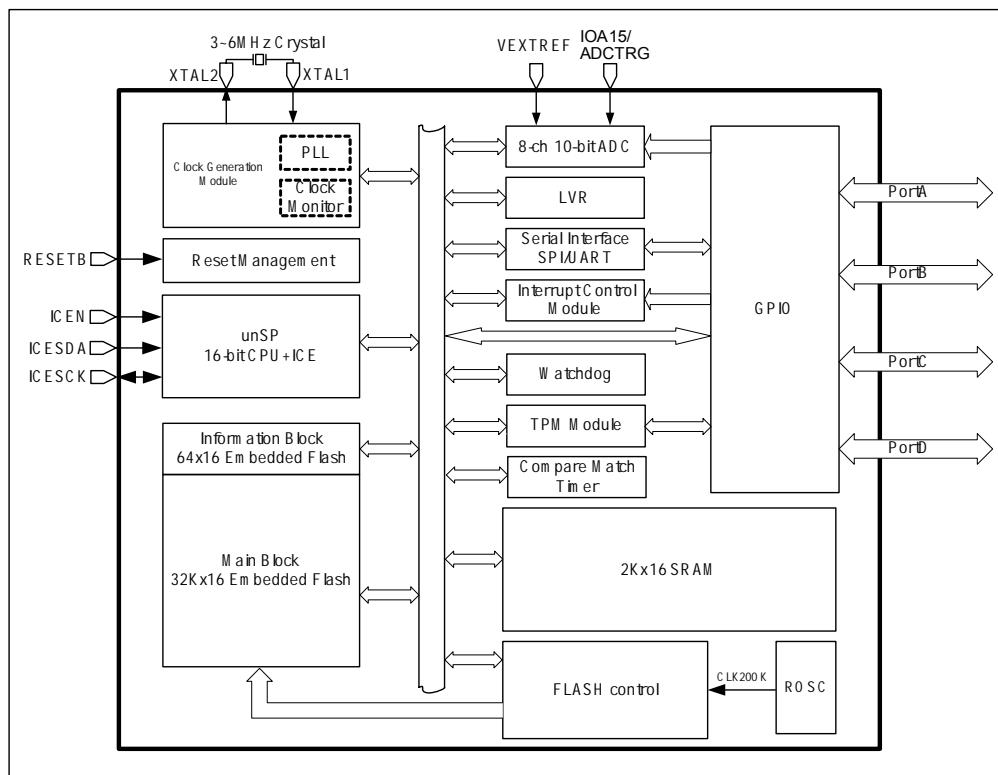


Figure 1-1 SPMC75F2413A function block diagram

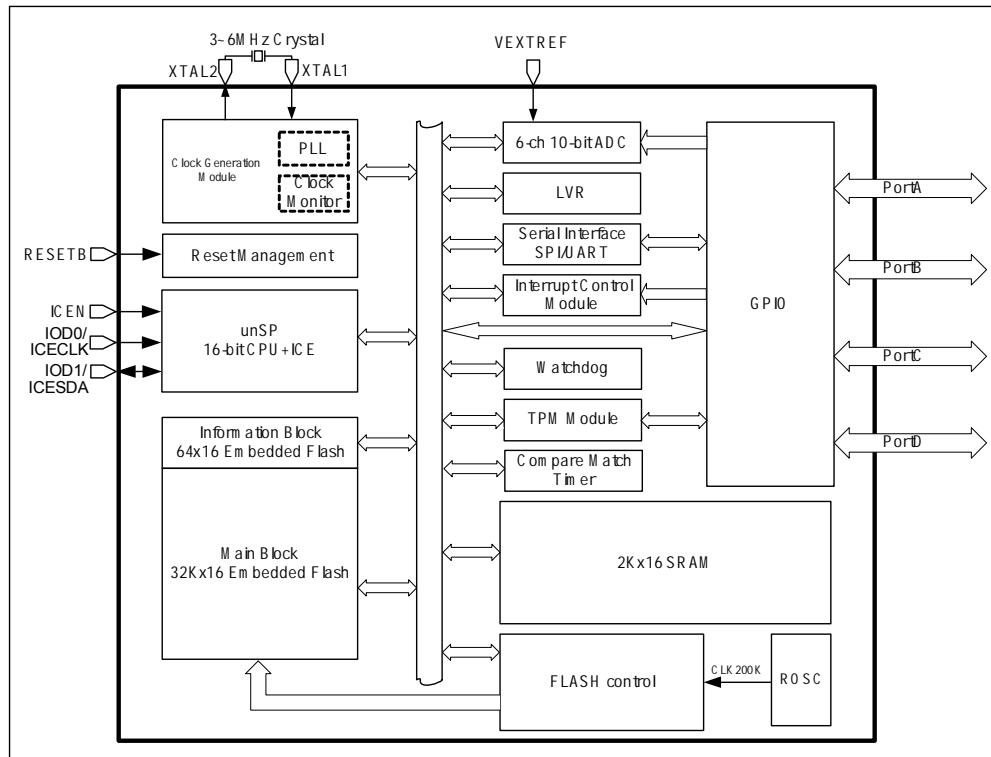


Figure 1-2 SPMC75F2313A function block diagram

1.7 Device Feature

- SUNPLUS 16-bit u'nSP processor (ISA 1.2)
- Operating voltage:
 - Core: 4.5V~ 5.5V
- Operating speed: Maximum 24MHz
- Operating temperature: -40 °C~85 °C
- On-chip Memory
 - 32KW (32K*16) Flash
 - 2KW (2K*16) SRAM
- On-chip PLL based clock generation
- Watchdog timer
- 10-bit analog-to-digital converter
 - 8 multiplexed input channels
 - 10us (100kHz) conversion time
- Serial communication interface
 - UART
 - SPI
- Up to 64 GPIO pins
- Power management
 - 2 power-down modes: Wait/Standby
 - Each peripheral can be powered down independently
- Two Compare Match Timers
- Five 16-bit general-purpose timers
 - 2 for PWM, 2 for rotor speed capturing, 1 for speed loop
 - MCP Timer channel 3 supports TIO3A-TIO3F, MCP Timer channel 4 supports TIO4A-TIO4F
 - PWM timers support up/down count, compare match output function
 - PDC Timer channel 0/1 each supports 3-channel capture input TIO0A-TIO0C and TIO1A-TIO1C
 - TPM Timer channel 2 supports Capture/PWM compare match output function
- Twelve 16-bit PWM outputs
 - 2-ch Motor drive PWM outputs (3-phase 6-pin complementary PWM outputs)
 - TIO3A-TIO3F work with MCP Timer channel 3, TIO4A-TIO4F work with MCP Timer channel 4
 - Center- or Edge-aligned PWM outputs
 - Emergency PWM outputs shutdown with external fault protection pins
 - Programmable Dead time control
 - PWM service and fault interrupt generation
 - Capable of driving AC induction and BLDC motors
- Embedded In-Circuit-Emulation Circuit

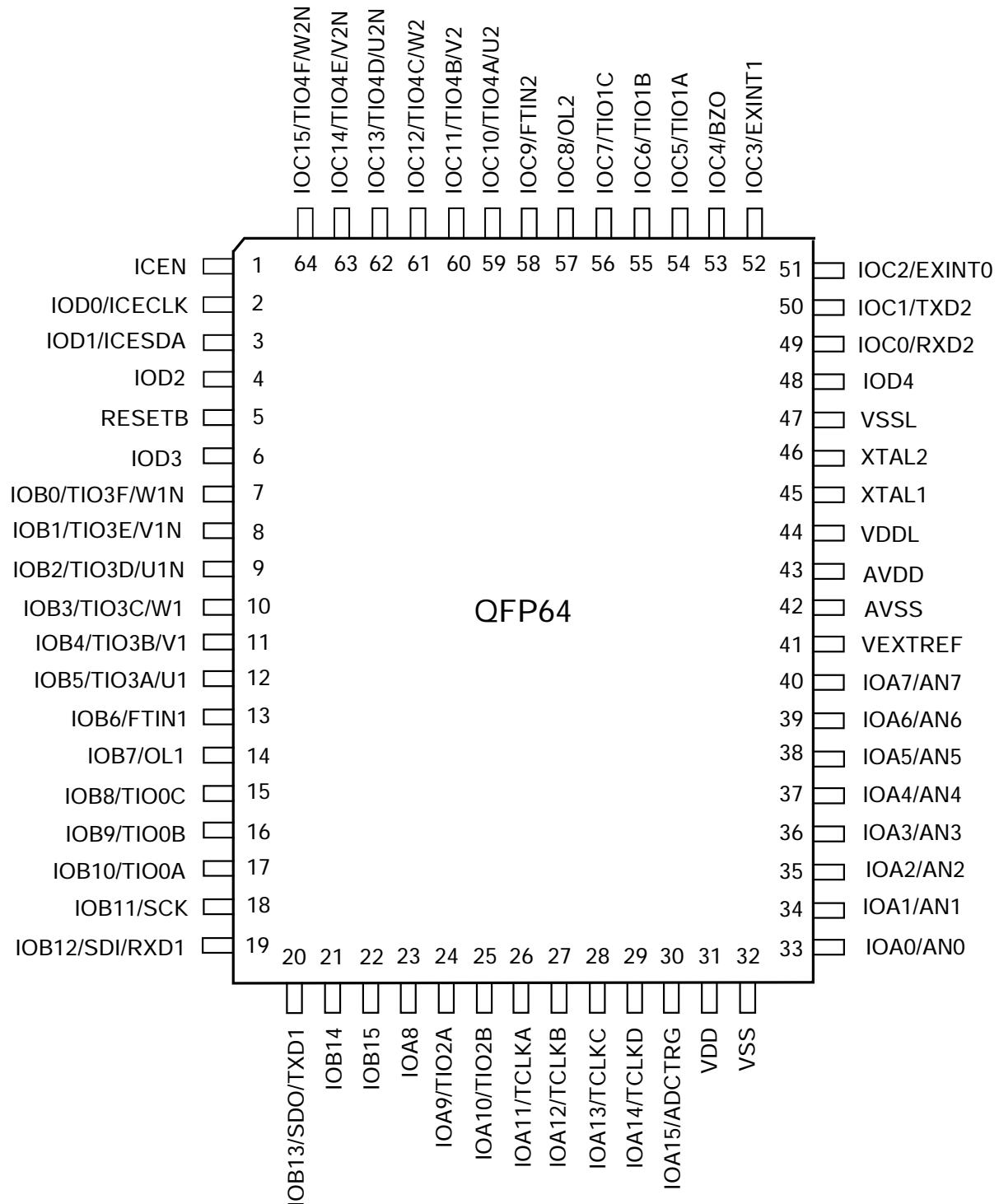
Table 2-1 shows the function outline of SPMC75X family MCU.

Table 2-1 The function outline of SPMC75X family MCU

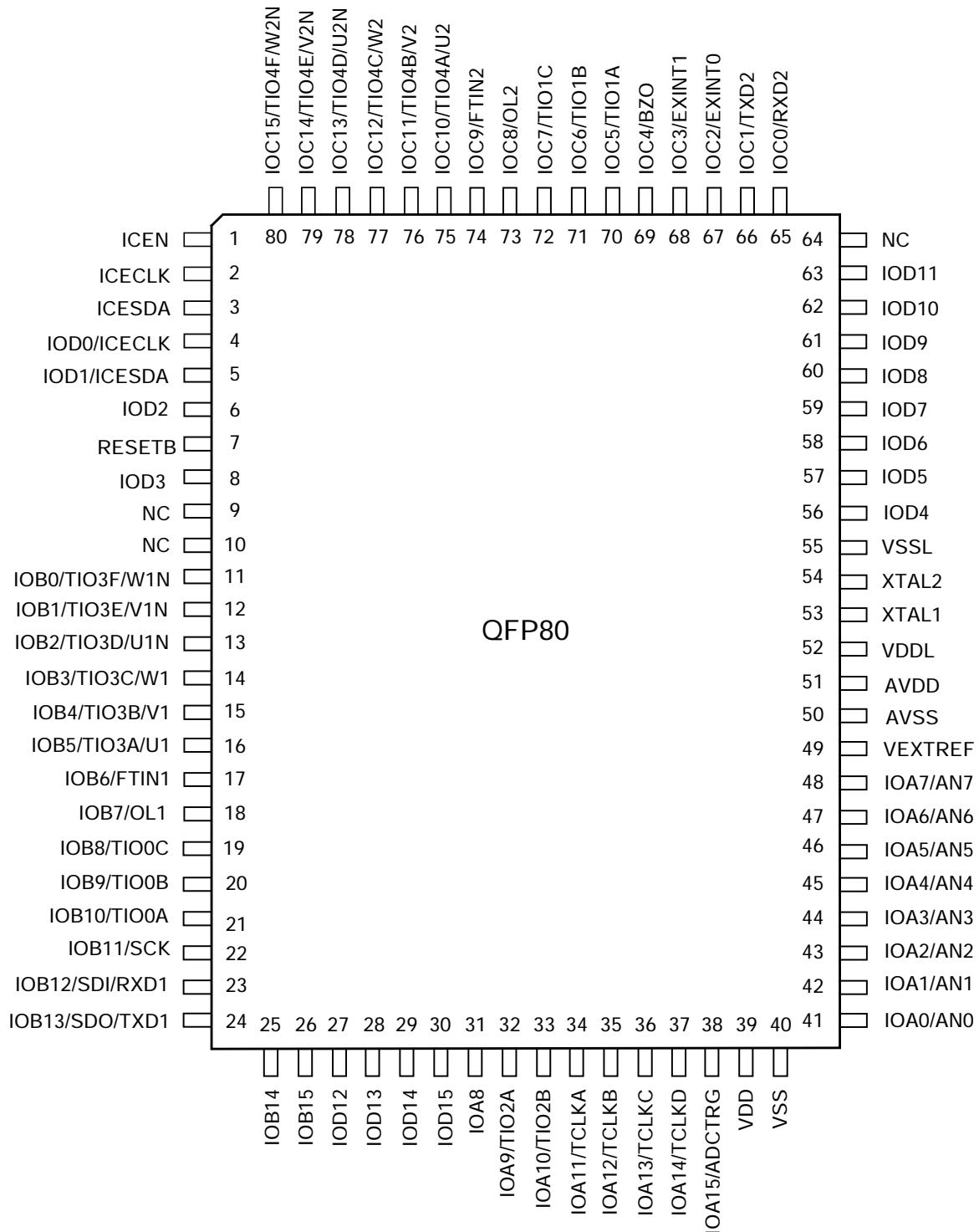
Function	Device	SPMC75F2313A	SPMC75F2413A
Internal	Program (Flash)	32K words	32K words
Memory	Data (SRAM)	2K words	2K words
Operating Speed		24 MHz	24 MHz
Operating Voltage		4.5 ~ 5.5V	4.5 ~ 5.5V
I/O ports (maximum)		33	64
Timers	PDC timers	16 bit x 2	16 bit x 2
	TPM timers	16 bit x 1	16 bit x 1
	MCP timers	16 bit x 1	16 bit x 2
	CMT timers	16 bit x 2	16 bit x 2
General PWM Output		16 bit x 8	16 bit x 8
Motor Control PWM Output		16 bit x 6	16 bit x 12
Input Capture		16 bit x 8	16 bit x 8
Position Detection Change Mode		Only for PDC 1 timer	Both PDC0 and PDC 1 timers
Phase Counting Mode		Only for PDC 1 timer	Both PDC0 and PDC 1 timers
A/D converter		10 bit resolution 6 channels	10 bit resolution 8 channels
SPI		Yes	Yes
UART		Yes, 300 ~ 115200 bps	Yes, 300 ~ 115200 bps
Watchdog Timer		Yes, 5.46 ~ 699.05 ms	Yes, 5.46 ~ 699.05 ms
Buzzer Output		No	1 channel 1.465 ~ 11.718 KHz
External Interrupt Inputs		No	Yes, 2 channels
Package		42 pin/SDIP 44 pin/LQFP	64 pin/QFP 80 pin/QFP

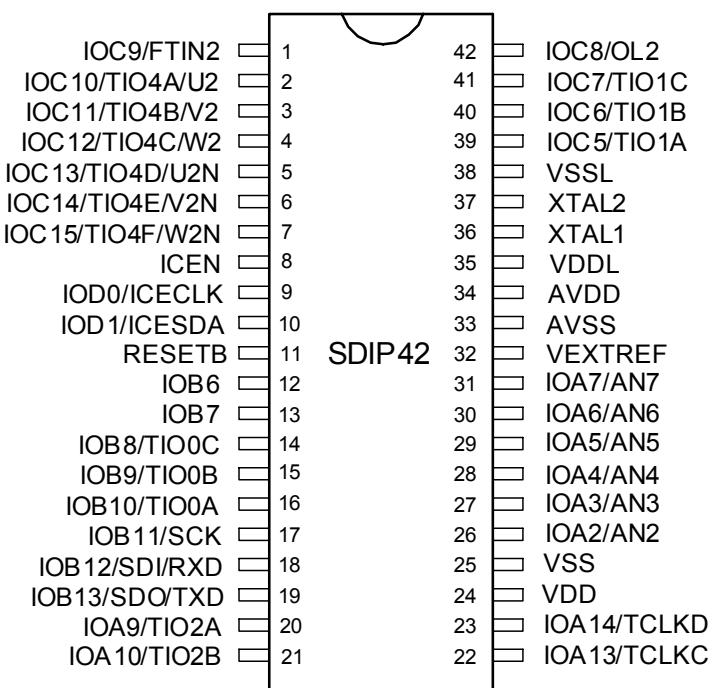
1.8 SPMC75X Family MCU Packages

1.8.1 SPMC75F2413A-PQ041(QFP64 Package, Pitch 1.0 mm)

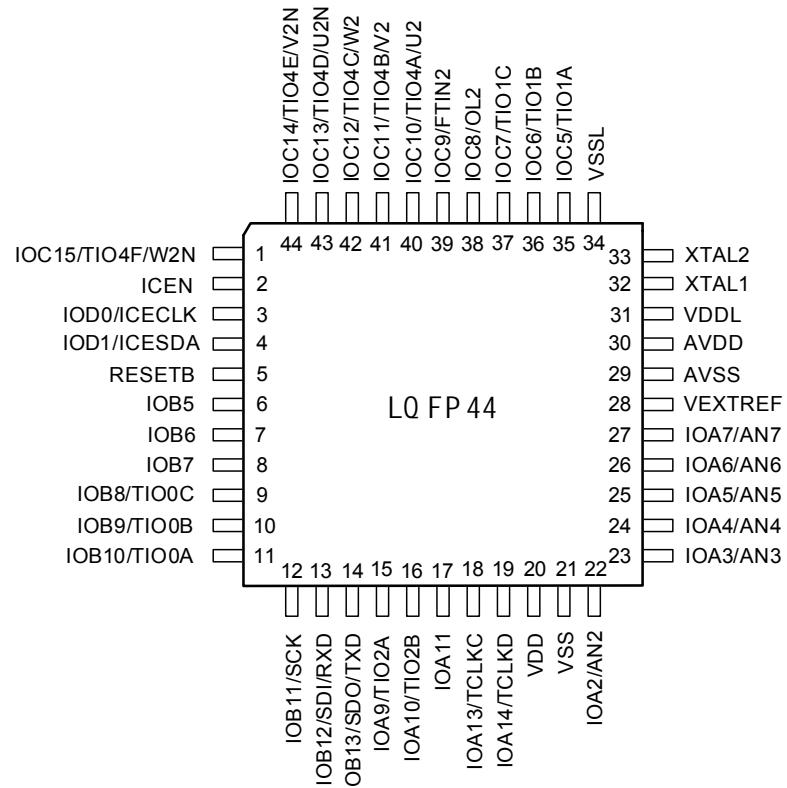


1.8.2 SPMC75F2413A-PQ051(QFP80 Package, Pitch 0.8 mm)



1.8.3 SPMC75F2313A-PD161(SDIP42 Package, Pitch 1.78 mm)

1.8.4 SPMC75F2313A-PL011(LQFP44 Package, Pitch 0.8 mm)



1.9 SPMC75X Family MCU Pin Descriptions

1.9.1 SPMC75F2413A 80-Pin QFP signals list

Mnemonic	PIN No	Type	Description
ICEN	1	I (PL)	ICE/Program or Normal mode control
ICECLK	2	I/O	ICE serial clock input (3V IO)
ICESDA	3	I/O	ICE serial address/data input/output (3V IO)
IOD0	4	I/O	IOD0
IOD1	5	I/O	IOD1
IOD2	6	I/O	IOD2
RESETB	7	I (PH)	External reset
IOD3	8	I/O	IOD3
NC	9	-	No Connection
NC	10	-	No Connection
IOB0/TIO3F/W1N	11	I/O	IOB0 or MCP channel 3 output F or motor drive W1N phase output
IOB1/TIO3E/V1N	12	I/O	IOB1 or MCP channel 3 output E or motor drive V1N phase output
IOB2/TIO3D/U1N	13	I/O	IOB2 or MCP channel 3 output D or motor drive U1N phase output
IOB3/TIO3C/W1	14	I/O	IOB3 or MCP channel 3 output C or motor drive W1 phase output
IOB4/TIO3B/V1	15	I/O	IOB4 or MCP channel 3 output B or motor drive V1 phase output
IOB5/TIO3A/U1	16	I/O	IOB5 or MCP channel 3 output A or motor drive U1 phase output
IOB6/FTIN1	17	I/O	IOB6 or external fault protection input 1
IOB7/OL1	18	I/O	IOB7 or overload protection input 1
IOB8/TIO0C	19	I/O	IOB8 or PDC channel 0 input/output C
IOB9/TIO0B	20	I/O	IOB9 or PDC channel 0 input/output B
IOB10/TIO0A	21	I/O	IOB10 or PDC channel 0 input/output A
IOB11/SCK	22	I/O	IOB11 or SPI clock input/output
IOB12/SDI/RXD1	23	I/O	IOB12 or SPI data input or UART receive data input 1
IOB13/SDO/TXD1	24	I/O	IOB13 or SPI data output or UART transmit data output 1
IOB14	25	I/O	IOB14
IOB15	26	I/O	IOB15
IOD12	27	I/O	IOD12
IOD13	28	I/O	IOD13
IOD14	29	I/O	IOD14
IOD15	30	I/O	IOD15
IOA8	31	I/O	IOA8
IOA9/TIO2A	32	I/O	IOA9 or TPM channel 2 input/output A
IOA10/TIO2B	33	I/O	IOA10 or TPM channel 2 input/output B
IOA11/TCLKA	34	I/O	IOA11 or external clock A input
IOA12/TCLKB	35	I/O	IOA12 or external clock B input
IOA13/TCLKC	36	I/O	IOA13 or external clock C input
IOA14/TCLKD	37	I/O	IOA14 or external clock D input
IOA15/ADCTRG	38	I/O	IOA15 or A/D converter external trigger to start a conversion
VDD	39	I	5V power input for IO and built-in regulator
VSS	40	I	Ground for IO

Mnemonic	PIN No	Type	Description
IOA0/AN0	41	I/O	IOA0 or analog input channel 0 of ADC
IOA1/AN1	42	I/O	IOA1 or analog input channel 1 of ADC
IOA2/AN2	43	I/O	IOA2 or analog input channel 2 of ADC
IOA3/AN3	44	I/O	IOA3 or analog input channel 3 of ADC
IOA4/AN4	45	I/O	IOA4 or analog input channel 4 of ADC
IOA5/AN5	46	I/O	IOA5 or analog input channel 5 of ADC
IOA6/AN6	47	I/O	IOA6 or analog input channel 6 of ADC
IOA7/AN7	48	I/O	IOA7 or analog input channel 7 of ADC
VEXTREF	49	I	ADC top voltage reference
AVSS	50	I	Analog ground for ADC
AVDD	51	I	Analog power for ADC
VDDL	52	O	External capacitance pin for internal step-down regulator/Digital power
XTAL1	53	I	External 3 ~ 6MHz crystal input
XTAL2	54	O	External 3 ~ 6MHz crystal output / External clock input
VSSL	55	I	Digital ground
IOD4	56	I/O	IOD4
IOD5	57	I/O	IOD5
IOD6	58	I/O	IOD6
IOD7	59	I/O	IOD7
IOD8	60	I/O	IOD8
IOD9	61	I/O	IOD9
IOD10	62	I/O	IOD10
IOD11	63	I/O	IOD11
NC	64	-	No connection
IOC0/RXD2	65	I/O	IOC0 or UART receive data input 2
IOC1/TXD2	66	I/O	IOC1 or UART transmit data output 2
IOC2/EXINT0	67	I/O	IOC2 or external interrupt input 0
IOC3/EXINT1	68	I/O	IOC3 or external interrupt input 1
IOC4/BZO	69	I/O	IOC4 or buzzer output
IOC5/TIO1A	70	I/O	IOC5 or PDC channel 1 input/output A
IOC6/TIO1B	71	I/O	IOC6 or PDC channel 1 input/output B
IOC7/TIO1C	72	I/O	IOC7 or PDC channel 1 input/output C
IOC8/OL2	73	I/O	IOC8 or overload protection input 2
IOC9/FTIN2	74	I/O	IOC9 or external fault input 2
IOC10/TIO4A/U2	75	I/O	IOC10 or MCP channel 4 output A or motor drive U2 phase output
IOC11/TIO4B/V2	76	I/O	IOC11 or MCP channel 4 output B or motor drive V2 phase output
IOC12/TIO4C/W2	77	I/O	IOC12 or MCP channel 4 output C or motor drive W2 phase output
IOC13/TIO4D/U2N	78	I/O	IOC13 or MCP channel 4 output D or motor drive U2N phase output
IOC14/TIO4E/V2N	79	I/O	IOC14 or MCP channel 4 output E or motor drive V2N phase output
IOC15/TIO4F/W2N	80	I/O	IOC15 or MCP channel 4 output F or motor drive W2N phase output

1.9.2 SPMC75F2413A 64-Pin QFP signals list

Mnemonic	PIN No	Type	Description
ICEN	1	I (PL)	ICE/Program or Normal mode control
IOD0/ICECLK	2	I/O	IOD0 or ICE serial clock input (for QFP64 package)
IOD1/ICESDA	3	I/O	IOD1 or ICE serial address/data input/output (for QFP64 package)
IOD2	4	I/O	IOD2
RESETB	5	I (PH)	External reset
IOD3	6	I/O	IOD3
IOB0/TIO3F/W1N	7	I/O	IOB0 or MCP channel 3 output F or motor drive W1N phase output
IOB1/TIO3E/V1N	8	I/O	IOB1 or MCP channel 3 output E or motor drive V1N phase output
IOB2/TIO3D/U1N	9	I/O	IOB2 or MCP channel 3 output D or motor drive U1N phase output
IOB3/TIO3C/W1	10	I/O	IOB3 or MCP channel 3 output C or motor drive W1 phase output
IOB4/TIO3B/V1	11	I/O	IOB4 or MCP channel 3 output B or motor drive V1 phase output
IOB5/TIO3A/U1	12	I/O	IOB5 or MCP channel 3 output A or motor drive U1 phase output
IOB6/FTIN1	13	I/O	IOB6 or external fault protection input 1
IOB7/OL1	14	I/O	IOB7 or overload protection input 1
IOB8/TIO0C	15	I/O	IOB8 or PDC channel 0 input/output C
IOB9/TIO0B	16	I/O	IOB9 or PDC channel 0 input/output B
IOB10/TIO0A	17	I/O	IOB10 or PDC channel 0 input/output A
IOB11/SCK	18	I/O	IOB11 or SPI clock input/output
IOB12/SDI/RXD1	19	I/O	IOB12 or SPI data input or UART receive data input 1
IOB13/SDO/TXD1	20	I/O	IOB13 or SPI data output or UART transmit data output 1
IOB14	21	I/O	IOB14
IOB15	22	I/O	IOB15
IOA8	23	I/O	IOA8
IOA9/TIO2A	24	I/O	IOA9 or TPM channel 2 input/output A
IOA10/TIO2B	25	I/O	IOA10 or TPM channel 2 input/output B
IOA11/TCLKA	26	I/O	IOA11 or external clock A input
IOA12/TCLKB	27	I/O	IOA12 or external clock B input
IOA13/TCLKC	28	I/O	IOA13 or external clock C input
IOA14/TCLKD	29	I/O	IOA14 or external clock D input
IOA15/ADCTRG	30	I/O	IOA15 or A/D converter external trigger to start a conversion
VDD	31	I	5V power input for IO and built-in regulator
VSS	32	I	Ground for IO
IOA0/AN0	33	I/O	IOA0 or analog input channel 0 of ADC
IOA1/AN1	34	I/O	IOA1 or analog input channel 1 of ADC
IOA2/AN2	35	I/O	IOA2 or analog input channel 2 of ADC
IOA3/AN3	36	I/O	IOA3 or analog input channel 3 of ADC
IOA4/AN4	37	I/O	IOA4 or analog input channel 4 of ADC
IOA5/AN5	38	I/O	IOA5 or analog input channel 5 of ADC
IOA6/AN6	39	I/O	IOA6 or analog input channel 6 of ADC
IOA7/AN7	40	I/O	IOA7 or analog input channel 7 of ADC
VEXTREF	41	I	ADC top voltage reference
AVSS	42	I	Analog ground for ADC
AVDD	43	I	Analog power for ADC

Mnemonic	PIN No	Type	Description
VDDL	44	O	External capacitance pin for internal step-down regulator/Digital power
XTAL1	45	I	External 3 ~ 6MHz crystal input
XTAL2	46	O	External 3 ~ 6MHz crystal output / External clock input
VSSL	47	I	Digital ground
IOD4	48	I/O	IOD4
IOC0/RXD2	49	I/O	IOC0 or UART receive data input 2
IOC1/TXD2	50	I/O	IOC1 or UART transmit data output 2
IOC2/EXINT0	51	I/O	IOC2 or external interrupt input 0
IOC3/EXINT1	52	I/O	IOC3 or external interrupt input 1
IOC4/BZO	53	I/O	IOC4 or buzzer output
IOC5/TIO1A	54	I/O	IOC5 or PDC channel 1 input/output A
IOC6/TIO1B	55	I/O	IOC6 or PDC channel 1 input/output B
IOC7/TIO1C	56	I/O	IOC7 or PDC channel 1 input/output C
IOC8/OL2	57	I/O	IOC8 or overload protection input 2
IOC9/FTIN2	58	I/O	IOC9 or external fault input 2
IOC10/TIO4A/U2	59	I/O	IOC10 or MCP channel 4 output A or motor drive U2 phase output
IOC11/TIO4B/V2	60	I/O	IOC11 or MCP channel 4 output B or motor drive V2 phase output
IOC12/TIO4C/W2	61	I/O	IOC12 or MCP channel 4 output C or motor drive W2 phase output
IOC13/TIO4D/U2N	62	I/O	IOC13 or MCP channel 4 output D or motor drive U2N phase output
IOC14/TIO4E/V2N	63	I/O	IOC14 or MCP channel 4 output E or motor drive V2N phase output
IOC15/TIO4F/W2N	64	I/O	IOC15 or MCP channel 4 output F or motor drive W2N phase output

1.9.3 SPMC75F2313A 42-Pin SDIP signals list

Mnemonic	PIN No.	Type	Description
IOC9/FTIN2	1	I/O	IOC9 or external fault input 2
IOC10/TIO4A/U2	2	I/O	IOC10 or MCP channel 4 output A or motor drive U2 phase output
IOC11/TIO4B/V2	3	I/O	IOC11 or MCP channel 4 output B or motor drive V2 phase output
IOC12/TIO4C/W2	4	I/O	IOC12 or MCP channel 4 output C or motor drive W2 phase output
IOC13/TIO4D/U2N	5	I/O	IOC13 or MCP channel 4 output D or motor drive U2N phase output
IOC14/TIO4E/V2N	6	I/O	IOC14 or MCP channel 4 output E or motor drive V2N phase output
IOC15/TIO4F/W2N	7	I/O	IOC15 or MCP channel 4 output F or motor drive W2N phase output
ICEN	8	I (PL)	ICE/Program or Normal mode control
IOD0/ICECLK	9	I/O	IOD0 or ICE serial clock input (for SDIP42 package)
IOD1/ICESDA	10	I/O	IOD1 or ICE serial address/data input/output (for SDIP42 package)
RESETB	11	I (PH)	External reset
IOB6	12	I/O	IOB6
IOB7	13	I/O	IOB7
IOB8/TIO0C	14	I/O	IOB8 or PDC channel 0 input/output C
IOB9/TIO0B	15	I/O	IOB9 or PDC channel 0 input/output B
IOB10/TIO0A	16	I/O	IOB10 or PDC channel 0 input/output A
IOB11/SCK	17	I/O	IOB11 or SPI clock input/output
IOB12/SDI/RXD1	18	I/O	IOB12 or SPI data input or UART receive data input 1
IOB13/SDO/TXD1	19	I/O	IOB13 or SPI data output or UART transmit data output 1
IOA9/TIO2A	20	I/O	IOA9 or TPM2 input/output A
IOA10/TIO2B	21	I/O	IOA10 or TPM2 input/output B
IOA13/TCLKC	22	I/O	IOA13 or external clock C input
IOA14/TCLKD	23	I/O	IOA14 or external clock D input
VDD	24	I	5V power input for IO
VSS	25	I	Ground for IO
IOA2/AN2	26	I/O	IOA2 or analog input channel 2 of ADC
IOA3/AN3	27	I/O	IOA3 or analog input channel 3 of ADC
IOA4/AN4	28	I/O	IOA4 or analog input channel 4 of ADC
IOA5/AN5	29	I/O	IOA5 or analog input channel 5 of ADC
IOA6/AN6	30	I/O	IOA6 or analog input channel 6 of ADC
IOA7/AN7	31	I/O	IOA7 or analog input channel 7 of ADC
VEXTREF	32	I	ADC top voltage reference
AVSS	33	I	Analog ground for ADC
AVDD	34	I	Analog power for ADC and built-in regulator
VDDL	35	O	External capacitance pin for internal step-down regulator/Digital power
XTAL1	36	I	External 3 ~ 6MHz crystal input for crystal oscillator
XTAL2	37	O	External 3 ~ 6MHz crystal output / External clock input
VSSL	38	I	Digital ground
IOC5/TIO1A	39	I/O	IOC5 or PDC channel 1 input/output A
IOC6/TIO1B	40	I/O	IOC6 or PDC channel 1 input/output B
IOC7/TIO1C	41	I/O	IOC7 or PDC channel 1 input/output C
IOC8/OL2	42	I/O	IOC8 or overload protection input 2

1.9.4 SPMC75F2313A 44-Pin LQFP signals list

Mnemonic	PIN No.	Type	Description
IOC15/TIO4F/W2N	1	I/O	IOC15 or MCP channel 4 output F or motor drive W2N phase output
ICEN	2	I (PL)	ICE/Program or Normal mode control
IOD0/ICECLK	3	I/O	IOD0 or ICE serial clock input (for LQFP44 package)
IOD1/ICESDA	4	I/O	IOD1 or ICE serial address/data input/output (for LQFP package)
RESETB	5	I (PH)	External reset
IOB5	6	I/O	IOB5
IOB6	7	I/O	IOB6
IOB7	8	I/O	IOB7
IOB8/TIO0C	9	I/O	IOB8 or PDC channel 0 input/output C
IOB9/TIO0B	10	I/O	IOB9 or PDC channel 0 input/output B
IOB10/TIO0A	11	I/O	IOB10 or PDC channel 0 input/output A
IOB11/SCK	12	I/O	IOB11 or SPI clock input/output
IOB12/SDI/RXD1	13	I/O	IOB12 or SPI data input or UART receive data input 1
IOB13/SDO/TXD1	14	I/O	IOB13 or SPI data output or UART transmit data output 1
IOA9/TIO2A	15	I/O	IOA9 or TPM 2 input/output A
IOA10/TIO2B	16	I/O	IOA10 or TPM 2 input/output B
IOA11	17	I/O	IOA11
IOA13/TCLKC	18	I/O	IOA13 or external clock C input
IOA14/TCLKD	19	I/O	IOA14 or external clock D input
VDD	20	I	5V power input for IO
VSS	21	I	Ground for IO
IOA2/AN2	22	I/O	IOA2 or analog input channel 2 of ADC
IOA3/AN3	23	I/O	IOA3 or analog input channel 3 of ADC
IOA4/AN4	24	I/O	IOA4 or analog input channel 4 of ADC
IOA5/AN5	25	I/O	IOA5 or analog input channel 5 of ADC
IOA6/AN6	26	I/O	IOA6 or analog input channel 6 of ADC
IOA7/AN7	27	I/O	IOA7 or analog input channel 7 of ADC
VEXTREF	28	I	ADC top voltage reference
AVSS	29	I	Analog ground for ADC
AVDD	30	I	Analog power for ADC and built-in regulator
VDDL	31	O	External capacitance pin for internal step-down regulator/Digital power
XTAL1	32	I	External 3 ~ 6MHz crystal input for crystal oscillator
XTAL2	33	O	External 3 ~ 6MHz crystal output / External clock input
VSSL	34	I	Digital ground
IOC5/TIO1A	35	I/O	IOC5 or PDC channel 1 input/output A
IOC6/TIO1B	36	I/O	IOC6 or PDC channel 1 input/output B
IOC7/TIO1C	37	I/O	IOC7 or PDC channel 1 input/output C
IOC8/OL2	38	I/O	IOC8 or overload protection input 2
IOC9/FTIN2	39	I/O	IOC9 or external fault input 2
IOC10/TIO4A/U2	40	I/O	IOC10 or MCP channel 4 output A or motor drive U2 phase output

Mnemonic	PIN No.	Type	Description			
IOC11/TIO4B/V2	41	I/O	IOC11 or MCP channel 4 output B or motor drive V2 phase output			
IOC12/TIO4C/W2	42	I/O	IOC12 or MCP channel 4 output C or motor drive W2 phase output			
IOC13/TIO4D/U2N	43	I/O	IOC13 or MCP channel 4 output D or motor drive U2N phase output			
IOC14/TIO4E/V2N	44	I/O	IOC14 or MCP channel 4 output E or motor drive V2N phase output			

Table 2-2 The pin signal descriptions of SPMC75X family MCU

Type : I = Input, O = Output, S = Supply

Pin Name	Package Type				Type	Main Function	Alternate Function	
	QFP80 (2413A)	QFP64 (2413A)	SDIP42 (2313A)	LQFP44 (2313A)				
VDD	39	31	24	20	S	+5V power input for IO and internal regulator		
VSS	40	32	25	21	S	Ground for IO		
AVSS	50	42	33	29	S	Analog ground for ADC		
AVDD	51	43	34	30	S	Analog power for ADC		
VDDL	52	44	35	31	S	External capacitance for voltage stable		
VEXTREF	49	41	32	28	S	ADC top voltage reference		
XTAL1	53	45	36	32	I	External 3 ~ 6 MHz crystal input		
XTAL2	54	46	37	33	O	External 3 ~ 6MHz crystal output / External clock input		
VSSL	55	47	38	34	I	Digital ground		
ICEN	1	1	8	2	I	ICE/Program or Normal mode control		
IOD0/ICECLK	2	2	9	3	I/O	ICE serial clock input (3V IO), independent IO function only in QFP80		
IOD1/ICESDA	3	3	10	4	I/O	ICE serial address/data input/output (3V IO), independent IO function only in QFP80		
RESETB	7	5	11	5	I	External reset		
IOA0/AN0	41	33	—	—	I/O	Port A0	ADC analog input channel 0	
IOA1/AN1	42	34	—	—	I/O	Port A1	ADC analog input channel 1	
IOA2/AN2	43	35	26	22	I/O	Port A2	ADC analog input channel 2	
IOA3/AN3	44	36	27	23	I/O	Port A3	ADC analog input channel 3	
IOA4/AN4	45	37	28	24	I/O	Port A4	ADC analog input channel 4	
IOA5/AN5	46	38	29	25	I/O	Port A5	ADC analog input channel 5	
IOA6/AN5	47	39	30	26	I/O	Port A6	ADC analog input channel 6	
IOA7/AN7	48	40	31	27	I/O	Port A7	ADC analog input channel 7	
IOA8	31	23	—	—	I/O	Port A8		
IOA9/TIO2A	32	24	20	15	I/O	Port A9	TPM channel 2 input/output A	
IOA10/TIO2B	33	25	21	16	I/O	Port A10	TPM channel 2 input/output B	
IOA11/TCLKA	34	26	—	17	I/O	Port A11	External clock A input (only 2413A)	
IOA12/TCLKB	35	27	—	—	I/O	Port A12	External clock B input (only 2413A)	
IOA13/TCLKC	36	28	22	18	I/O	Port A13	External clock C input	

Pin Name	Package Type				Type	Main Function	Alternate Function
	QFP80 (2413A)	QFP64 (2413A)	SDIP42 (2313A)	LQFP44 (2313A)			
IOA14/TCLKD	37	29	23	19	I/O	Port A14	External clock D input
IOA15/ADCTRG	38	30	—	—	I/O	Port A15	ADC external trigger (only 2413A)
IOB0/TIO3F/W1N	11	7	—	—	I/O	Port B0	MCP channel 3 output F or motor drive W1N phase output (only 2413A)
IOB1/TIO3E/V1N	12	8	—	—	I/O	Port B1	MCP channel 3 output E or motor drive V1N phase output (only 2413A)
IOB2/TIO3D/U1N	13	9	—	—	I/O	Port B2	MCP channel 3 output D or motor drive U1N phase output (only 2413A)
IOB3/TIO3C/W1	14	10	—	—	I/O	Port B3	MCP channel 3 output C or motor drive W1 phase output (only 2413A)
IOB4/TIO3B/V1	15	11	—	—	I/O	Port B4	MCP channel 3 output B or motor drive V1 phase output (only 2413A)
IOB5/TIO3A/U1	16	12	—	6	I/O	Port B5	MCP channel 3 output A or motor drive U1 phase output (only 2413A)
IOB6/FTIN1	17	13	12	7	I/O	Port B6	External fault protection input 1 (only 2413A)
IOB7/OL1	18	14	13	8	I/O	Port B7	Overload protection input 1 (only 2413A)
IOB8/TIO0C	19	15	14	9	I/O	Port B8	PDC channel 0 input/output C
IOB9/TIO0B	20	16	15	10	I/O	Port B9	PDC channel 0 input/output B
IOB10/TIO0A	21	17	16	11	I/O	Port B10	PDC channel 0 input/output A
IOB11/SCK	22	18	17	12	I/O	Port B11	SPI clock input/output
IOB12/SDI/RXD1	23	19	18	13	I/O	Port B12	SPI data input or UART RXD 1
IOB13/SDO/TXD1	24	20	19	14	I/O	Port B13	SPI data output or UART TXD 1
IOB14	25	21	—	—	I/O	Port B14 (only 2413A)	
IOB15	26	22	—	—	I/O	Port B15 (only 2413A)	
IOC0/RXD2	65	49	—	—	I/O	Port C0	UART RXD 2 (only 2413A)
IOC1/TXD2	66	50	—	—	I/O	Port C1	UART TXD 2 (only 2413A)
IOC2/EXINT0	67	51	—	—	I/O	Port C2	External interrupt input 1 (only 2413A)
IOC3/EXINT1	68	52	—	—	I/O	Port C3	External interrupt input 2 (only 2413A)
IOC4/BZO	69	53	—	—	I/O	Port C4	Buzzer output (only 2413A)
IOC5/TIO1A	70	54	39	35	I/O	Port C5	PDC channel 1 input/output A
IOC6/TIO1B	71	55	40	36	I/O	Port C6	PDC channel 1 input/output B
IOC7/TIO1C	72	56	41	37	I/O	Port C7	PDC channel 1 input/output C
IOC8/OL2	73	57	42	38	I/O	Port C8	Overload fault input 2
IOC9/FTIN2	74	58	1	39	I/O	Port C9	External fault input 2
IOC10/TIO4A/U2	75	59	2	40	I/O	Port C10	MCP channel 4 output A or motor drive U2 phase output

Pin Name	Package Type				Type	Main Function	Alternate Function
	QFP80 (2413A)	QFP64 (2413A)	SDIP42 (2313A)	LQFP44 (2313A)			
IOC11/TIO4B/V2	76	60	3	41	I/O	Port C11	MCP channel 4 output B or motor drive V2 phase output
IOC12/TIO4C/W2	77	61	4	42	I/O	Port C12	MCP channel 4 output C or motor drive W2 phase output
IOC13/TIO4D/U2N	78	62	5	43	I/O	Port C13	MCP channel 4 output D or motor drive U2N phase output
IOC14/TIO4E/V2N	79	63	6	44	I/O	Port C14	MCP channel 4 output E or motor drive V2N phase output
IOC15/TIO4F/W2N	80	64	7	1	I/O	Port C15	MCP channel 4 output F or motor drive W2N phase output
IOD2	6	4	—	—	I/O	Port D2	
IOD3	8	6	—	—	I/O	Port D3	
IOD4	56	48	—	—	I/O	Port D4	
IOD5	57	—	—	—	I/O	Port D5	
IOD6	58	—	—	—	I/O	Port D6	
IOD7	59	—	—	—	I/O	Port D7	
IOD8	60	—	—	—	I/O	Port D8	
IOD9	61	—	—	—	I/O	Port D9	
IOD10	62	—	—	—	I/O	Port D10	
IOD11	63	—	—	—	I/O	Port D11	
IOD12	27	—	—	—	I/O	Port D12	
IOD13	28	—	—	—	I/O	Port D13	
IOD14	29	—	—	—	I/O	Port D14	
IOD15	30	—	—	—	I/O	Port D15	

1.10 Development Support

SUNPLUS offers a wide range of development tools that allow users to efficiently develop and debug application code. All tools developed by SUNPLUS operate under the un'SP® Integrated Development Environment (IDE). This IDE supports all the follow functions:

1. Code generation
2. Software debug
3. Device programmer
4. Product evaluation boards

While using IDE to simulate SPMC75F2413A, the IDE must be set to SPMC75F2413A or SPMC75F2313A. This body includes ICE and all functions of SPMC75F2413A and SPMC75F2313A..

2 Oscillator

2.1 Oscillator

Only one oscillator systems in the SPMC75X family microcontroller, providing clocks for operation. The oscillator system must be use crystal or resonator device for clock generation. The range is between 3M ~ 6MHz. The oscillator system input will be as PLL (Phase-Lock-Loop) clock source, and the PLL circuit pumps the input clock to four times. Thus, if the crystal connected 6Mhz, the clock will be pumped to 24Mhz.

The SPMC75X family also has an internal clock source for flash circuit. The system will generate a 1600KHz by internal R-oscillator and produce 200KHz clock by internal divider.

The on-chip oscillator circuits are listed as follows:

- 1600KHz R-oscillator (generating 200KHZ clock by divider)
- Phase-Lock Loop (pumps crystal frequency to four times)
- crystal input & output pads

2.1.1 200KHz R-Oscillator

The 1600KHz R-oscillator provides a 200KHz clock for flashing controller to generate the necessary control signals meeting the timing specifications of flash erasing and programming. It can be disabled in either sleep mode or standby mode for power saving.

2.1.2 Phase-lock Loop

There is an on-chip PLL (Phase-lock loop) circuit available for generating system clock. This module provides all necessary clock signals for the device.

The clock generation module provides two modes of operation:

- PLL bypassed operation

This mode allows the use of an external clock input to provide the system clock to the device.

- PLL operation

This mode allows the use of the internal PLL circuit to provide the system clock to the device. The PLL takes a reference clock for generating system clock. On-chip crystal/resonator oscillator clock or external clock input can be selected as the reference clock of PLL circuit.

The PLL output clock rate is four times of reference clock. When power-on or system reset or wake-up from standby, CPU will halt 16384 oscillator reference clocks (F_{IN}) for oscillator and PLL stable. The stable time is about 82ms when reference clock is 200KHz.

2.1.3 Clock Monitoring

A clock monitoring circuit is also available to detect whether the crystal and system clock run normally. If a clock halt is detected, the twelve motor drive PWM pins (TIO3A~F and TIO4A~F) will be set to high-impedance state, regardless their pin function settings, and an interrupt will be issued to notify CPU.

Figure 3-1 shows the timing waveform when clock source is detected at fail condition.

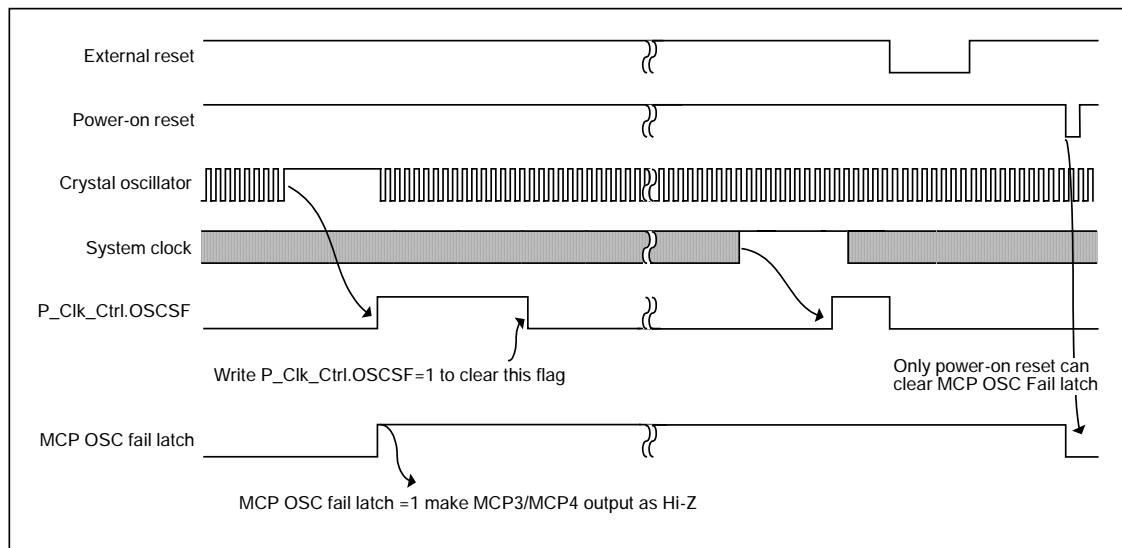


Figure 2-1 clock source fail condition timing waveform

2.2 Function Block

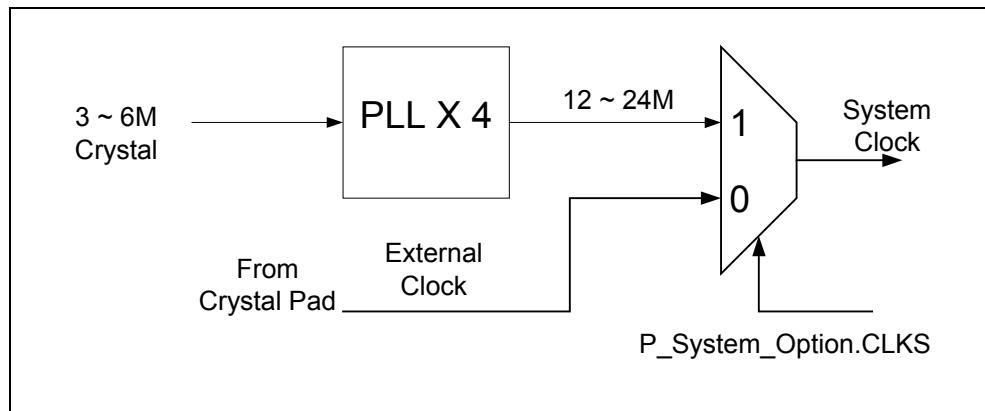


Figure 2-2 Function Block

2.3 System Clock Control Register

2.3.1 P_Clk_Ctrl (0x7007): System Clock Control Register

This register is used for monitoring CPU clock status.

B15	B14	B13	B2	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
OSCSF	OSCIE	Reserved					
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Bit 15 **OSCSF**: Oscillator status flag. This flag indicates whether the on-chip crystal oscillator or external input clock runs normally. If the oscillator or external clock input stopped, this bit will be set to '1'. Write this bit to '1' will clear this flag.

Read 0 = Oscillator operates normally

Read 1 = Oscillator failed

Write 1 = Clear this flag

Bit 14 **OSCIE**: Oscillator fail interrupt enable bit

0 = Disable

1 = Enable

Bit 13:0 Reserved

2.4 Application Circuit

There are three application circuit of oscillator in SPMC75X family MCU.

(1) Crystal input

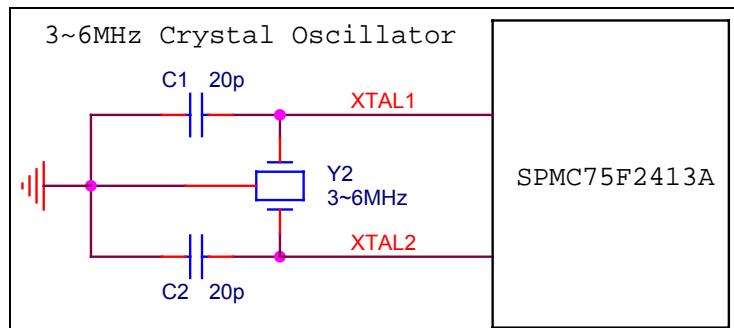


Figure 2-3 crystal circuit connection

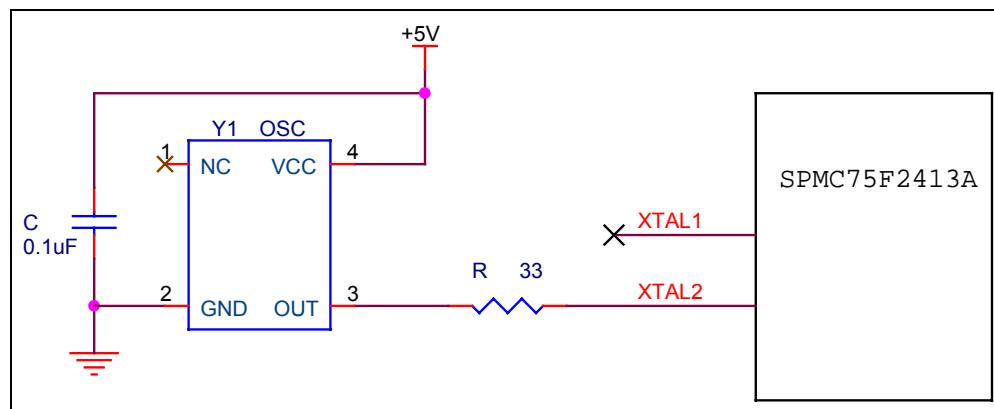
(2) Oscillator input


Figure 2-4 oscillator circuit connection

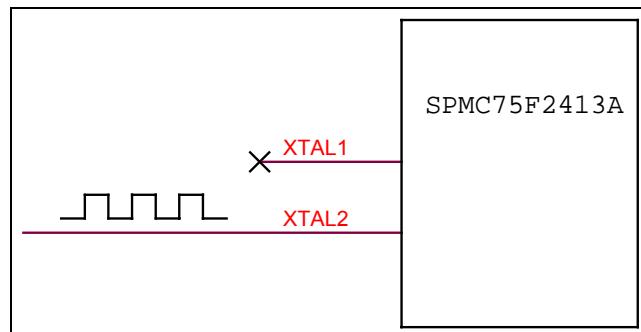
(3) External clock source input


Figure 2-5 external clock source connection

2.5 Design Tips

Enable Oscillator fail interrupt:

```

P_Clk_Ctrl->B.OSCIE = CB_CLK_OSCIE;           /*Enable Oscillator Interrupt */

/* Check Oscillator Status Flag & Clear Oscillator Status Flag */
if(P_Clk_Ctrl->B.OSCSF & CB_CLK_OSCSF) /* Check Oscillator status flag */
    P_Clk_Ctrl->B.OSCSF = CB_CLK_OSCSF; /*Clear Oscillator status flag */
    
```

Listing3-1 OSF interrupt design tips

3 Reset

3.1 Introduction

In SPMC75X family MCU, the reset logic is used for leading MCU into a known state. The source of reset can be determined by using the device status bits. The reset circuit can be used for increasing system reliability.

The various types of resets:

- 1 . Power on reset (POR)
- 2 . External reset
- 3 . Low voltage reset (LVR)
- 4 . Watchdog timer reset (WDTR)
- 5 . Illegal address reset (IAR)
 - Accessing invalid addresses (0x0800-0x6FFF, 0x7500-0x7FFF)
 - Write invalid values to 0x700C, 0x700E
- 6 . Illegal instruction reset(IIS)
 - An invalid instruction is decoded by CPU

3.2 Power-up Procedure

When power is turned on, option bits are read by the system. The option bits are stored in the first word of embedded flash information block (address = 0x8000). When power is turned on, the system reset is activated until the power-on-timer counts 16384 cycles of 200KHz clock then, reset signal is released.

Remarkably, all GPIO is on the high impedance state initially and can be configured after power-on procedure. Power-up procedure timing is shown in Figure 3-1.

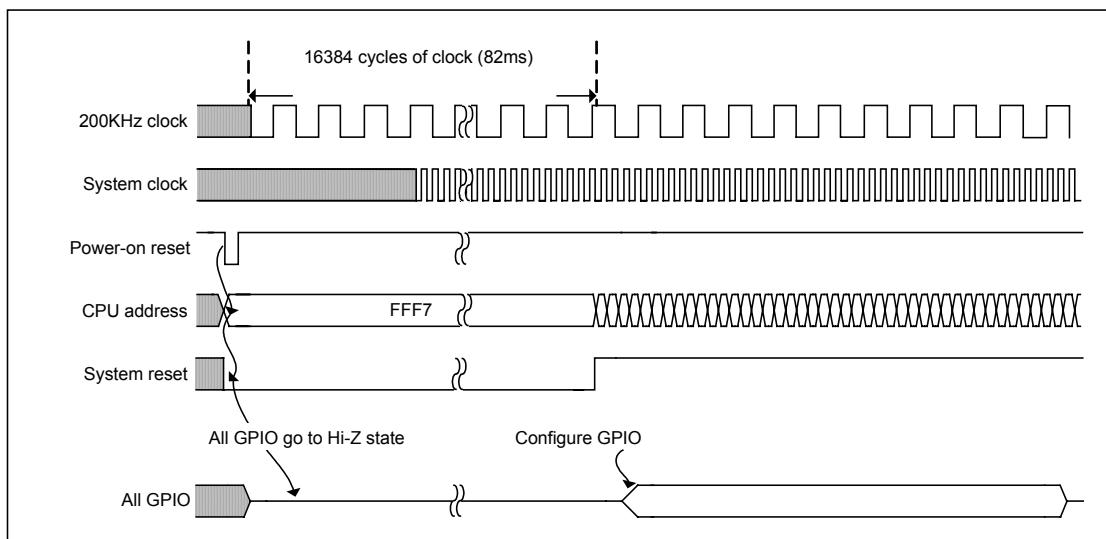


Figure 3-1 Power-up procedure

3.3 Reset Mode

3.3.1 Power on reset (POR)

A power-on-reset (POR) is generated when VDD rising is detected. When the rising rate is greater than 0.5 V per us and VDD raised to acceptable level, the power on reset circuit starts working. In SPMC75X family MCU, the power-up timer will counts for 16384 X'TAL clock ticks when power on reset. After a reset time, all registers will be initiated at designate value. See Figure 4-2 for timing details.

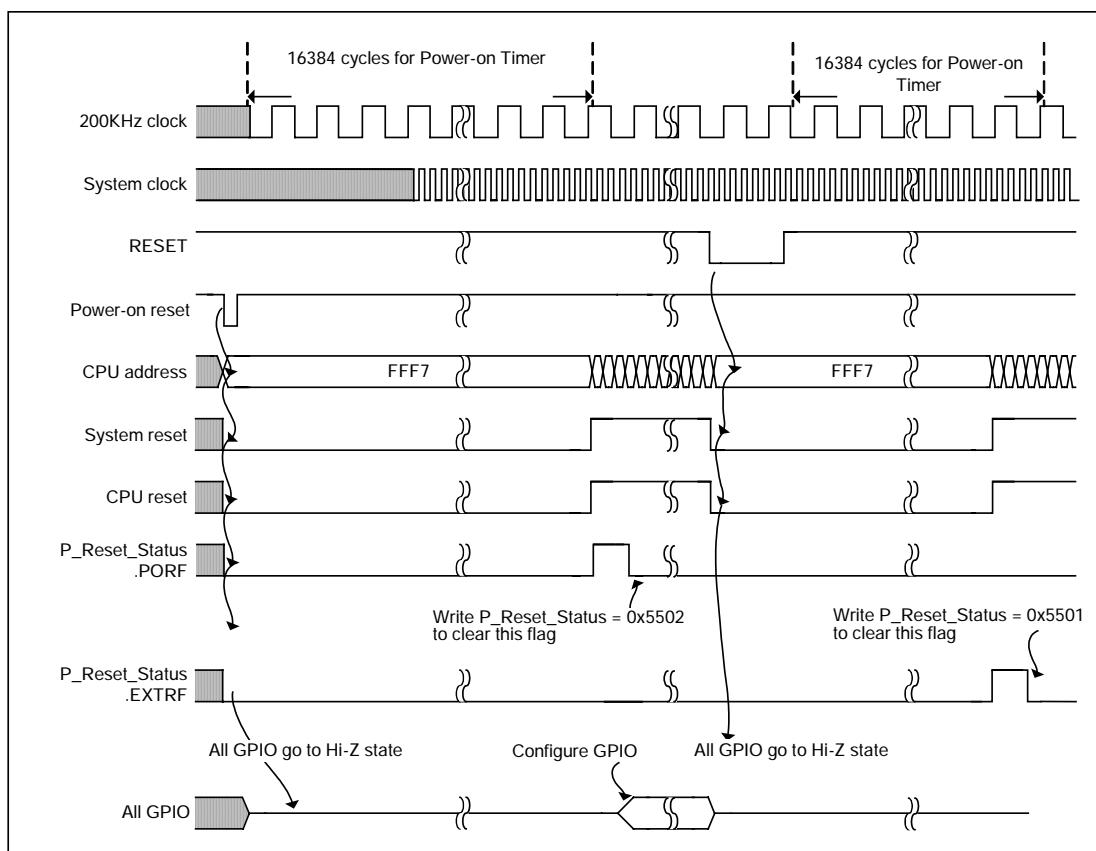


Figure 3-2 External reset, Power-On Reset, Power-Up Timer Timing

3.3.2 External Reset

The SPMC75X family MCU provides an external pin to force the system returning to the initial status. The RESET is used to connect an RC circuit, shown in Figure 4-2. This pin is a low active signal. When the RESETB pin falls below $0.3 \times VDD$, system will be forced entering into reset state. If the recharging voltage on capacitor is greater than acceptable voltage, system reset will last 16384 crystal clocks when power on reset and then complete the whole reset function. See Figure 4-3 for timing information.

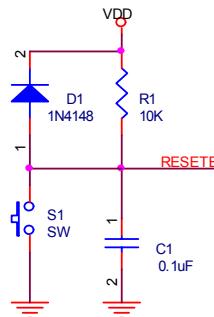


Figure 3-3 External Reset Circuit

3.3.3 Low Voltage Reset (LVR)

The low voltage reset (LVR) function is available in the device for more reliable operation. When power supply voltage drops below 4.09V, a low voltage reset will be issued. When LVR is asserted, a system reset will be generated. CPU and all peripherals will be reset. When supply voltage up to 4.19V, system will leave reset status. The Figure 4-4 shows the typical low voltage reset timing.

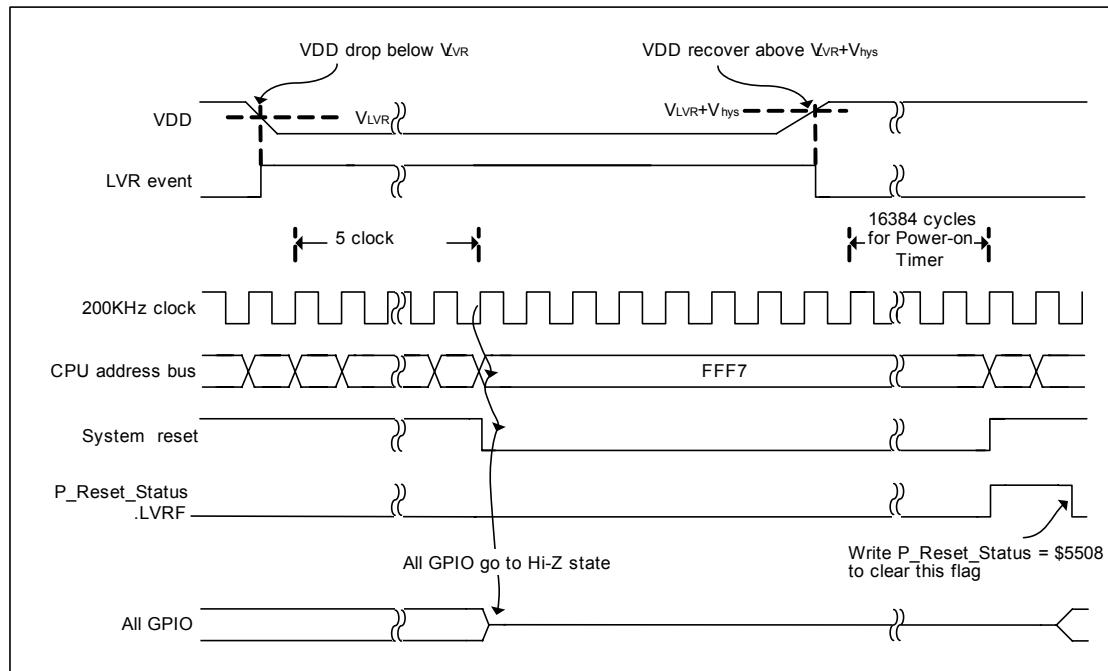


Figure 3-4 Low voltage reset timing

3.3.4 Watch Dog Timer Reset (WDTR)

On-chip watchdog circuitry makes the device entering into reset when the MCU goes into unknown state and without any watchdog clearance. This function ensures the MCU does not continue to work in abnormal condition. The WDTR can be configured by the B1 of **P_System_Option (0x8000)** and the B0 of **P_WatchDog_Ctrl(0x700A)**. We can set the overflow timer by using B[3:1] of **P_WatchDog_Ctrl**. Also, when “0xA005” is written into **P_WatchDog_Clr(W) (0x700B)**, the watchdog timer will be reset and continue to count. If **P_WatchDog_Clr** is not written between watchdog counting interval, the system will

be forced to reset CPU. The watchdog reset is disabled in ICE mode. Figure 4-5 shows the watchdog timer reset timing waveform.

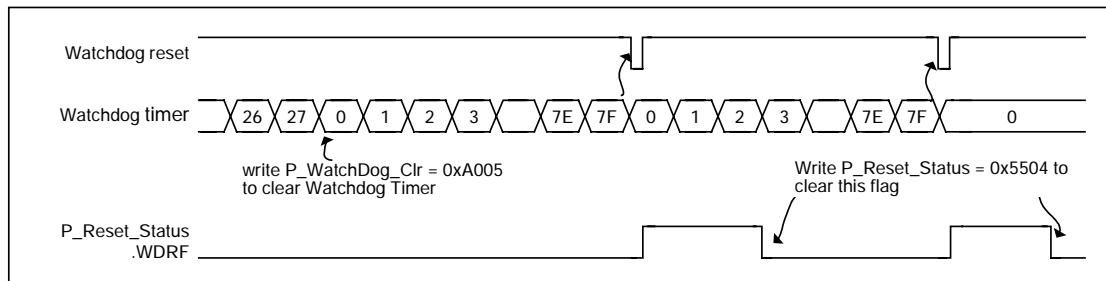


Figure 3-5 Watchdog timer reset timing

3.3.5 Illegal address reset (IAR)

The SPMC75X family MCU offers an illegal address reset for preventing system entering into illegal address. When system goes into illegal address, only CPU will be reset.

The invalid addresses are between from 0x0800 to 0x6FFF and from 0x7600 to 0x7FFF. So and instruction goes into these area, system will create a reset signal to reset CPU.

Another area if invalid values write into 0x700C or 0x700E the system will create a reset signal to reset CPU. Refer to Figure 4-6 to see the IAR reset timing details.

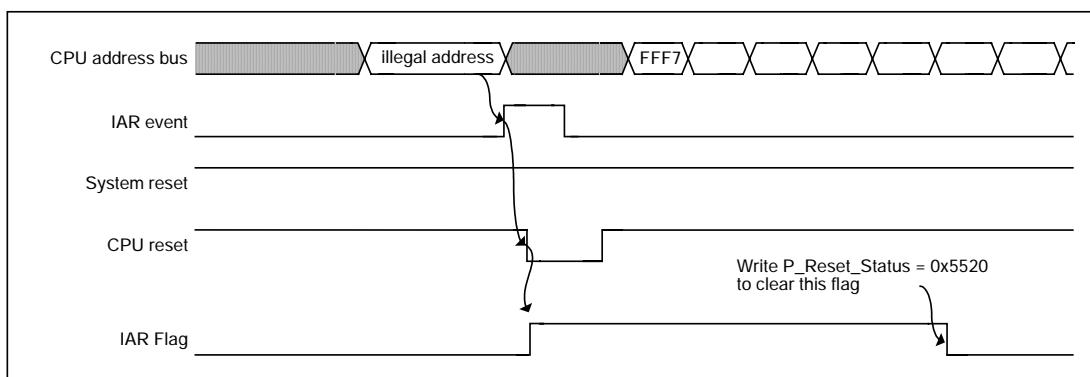


Figure 3-6 Illegal address reset timing

3.3.6 Illegal instruction reset (IIR)

An invalid instruction is decoded by CPU. When system decodes the illegal instruction, only CPU will be reset and set reset P_Reset_Status.IIRF bit. User can use this bit to check system reset from illegal instruction and Figure 4-7 depicts the IIR reset timing waveform.

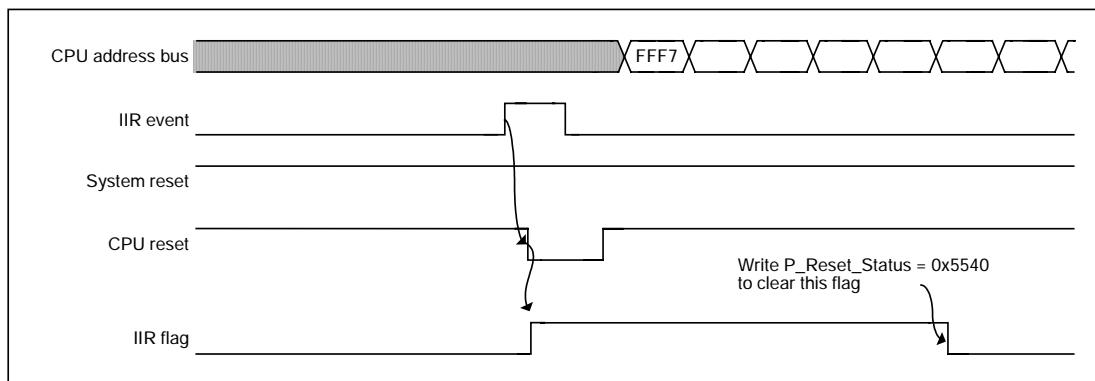


Figure 3-7 Illegal instruction reset timing

3.4 Reset Source Table

These reset sources can reset various modules in SPMC75X family MCU, see the following:

Reset Source	CPU	Peripheral
External Reset Pin	V	V
Power-on Reset	V	V
Watchdog Reset	V	Option
Low Voltage Reset	V	V
Illegal Address Reset	V	-
Illegal Instruction Reset	V	-

Note: Flash controller and other modules are reset by power on reset and external reset, which the reset signal retains until power on timer is counted over.

3.5 Control Register

3.5.1 P_Reset_Status(0x7006): Reset Status Register

This register shows the flag of reset status for firmware checking.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FCHK							

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved	IIRF	IARF	Reserved	LVRF	WDRF	PORF	EXTRF

Bit 15:8 **FCHK** : Flag clear check bits pattern

To properly clear reset flags, these bits must be written to '0x55'. Otherwise, the flags will not be cleared. These bits will be read as '0'.

Bit 7 Reserved

- BBit 6 **IIRF:** Illegal instruction reset flag. This bit indicates whether the Illegal Instruction Reset has occurred.
0 = Not occurred
1 = Occurred
This flag is cleared by writing '1' to this bit
- Bit 5 **IARF:** Illegal address reset flag. This bit indicates whether the Illegal Address Reset has occurred.
0 = Not occurred
1 = Occurred
This flag is cleared by writing '1' to this bit
- Bit 4 Reserved
- Bit 3 **LVRF:** Low voltage reset flag. This bit indicates whether the Low Voltage Reset has occurred.
0 = Not occurred
1 = Occurred
This flag is cleared by writing '1' to this bit
- Bit 2 **WDRF:** Watchdog reset flag. This bit indicates whether the Watchdog Reset has occurred.
0 = Not occurred
1 = Occurred
This flag is cleared by writing '1' to this bit
- Bit 1 **PORF:** Power-on reset flag. This bit indicates whether the Power-on Reset has occurred.
0 = Not occurred
1 = Occurred
This flag is cleared by writing '1' to this bit
- Bit 0 **EXTRF:** External reset pin reset flag. This bit indicates whether the External Pin Reset has occurred.
0 = Not occurred
1 = Occurred
This flag is cleared by writing '1' to this bit

3.6 Design Tips

3.6.1 Reset Status

The reset status will show the type of reset. User can use this register to check the reset status. The system can recover the condition before reset if the software is programmed properly. User can add the checking procedure at the beginning of program as follows.

【Example4-3】:

```
if(P_Reset_Status->B.IARF & CB_CLEAR_IARF)          /* Check IARF Reset Flag */  
{  
    P_Reset_Status->W = (0x5500 | CW_CLEAR_IARF); /* Clear IARF Reset Flag */  
}
```

Listing 4-1 IAR reset source design tips

4 MCU

In SPMC75X family MCU, the kernel of MCU is u'nSP version 1.2 defined by SUNPLUS and 16-bit microcontroller with some DSP functions. The functions include:

1:16-bit data bus / 22-bit address bus

- a: 4M words (8M bytes) memory space
- b: 64 banks / 64k words per bank

2:Thirteen 16-bit registers

- a: 4 general registers (R1-R4)
- b: 4 secondary registers (SR1-SR4)
- c: 4 system registers (SP, BR, SR, PC)
- d: inner registers (FR)

3:Ten interrupts

- a: 1 fast interrupt (FIQ)
- b: 8 normal interrupts (IRQ0-IRQ7)
- c: 1 software interrupt (BREAK)
- d: Support IRQ nested mode

4:Six address modes

- a: Immediate (I6/I16) : ALU operations with 6-bits/16-bits immediate value.
- b: Direct (A6/A16) : ALU operations with 6-bits/16bits direct memory addressing.
- c: Indirect : ALU operations with indirect memory addressing also for bit operation..
- d: Displacement : ALU operations with 6-bits displacement memory addressing.
- e: Multi-indirect (PUSH/POP)
- f: Register

5:16x16 multiplication & up to 16-level inner product operation

- a: Three multiplication mode: signed x signed, signed x unsigned, unsigned x unsigned
- b: 4 bits guard bit of inner product operation to avoid overflow
- c: Integer/Fraction mode

6:Non-Restoring Division

- a: 32-bits dividend and 16-bits divisor
- b: Need 16 continuous operation (DIVS, DIVQ) to generate correct quotient

7:Effective-exponent detect operation (EXP)**8:Bit operation**

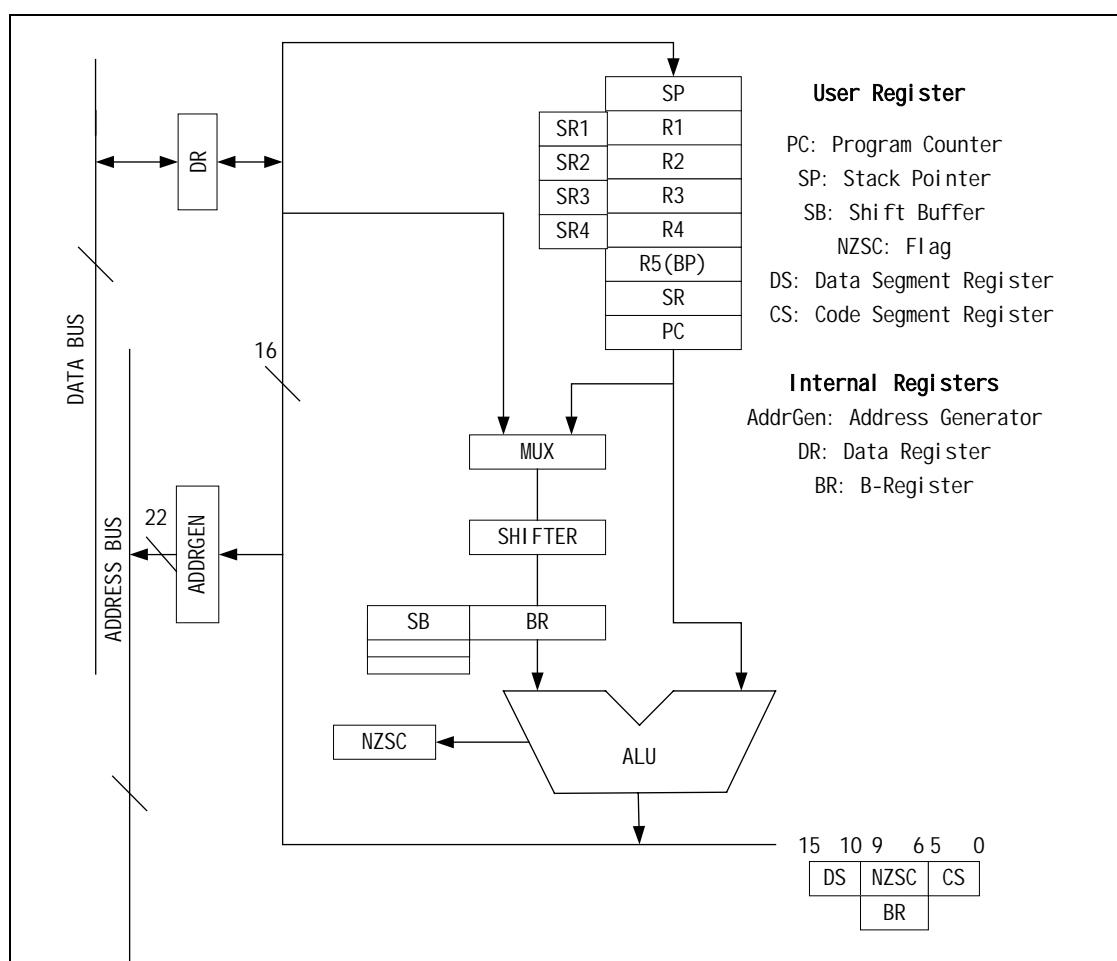
- a: Bit test / set / clearn/ inverse operation
- b: Destination can be register or memory.

9:Multi-cycles 16-bit shift operation

a: Support 32-bit shift with combining 2 shift instructions

10:Far Indirect JMP by MR register
11:Far Indirect Call by MR register
12:NOP operation
13: DS segment access instructions
14: CPU inner flags access instructions
15: Power Down/Sleep Mode

4.1 Function Block



4.2 CPU Instruction and Executions Cycle

The following table 5-1 shows all instruction types, operation styles, and execution cycles. The contents of table means:

1. **RW**: Memory Read Waiting cycle, RW = 0 if no wait state insertion and RW = N if wait state = N.
2. **SW**: Memory Write Waiting Cycle, SW= 0, except flash programming
3. **SRW** : store or read waiting cycle, SRW = SW when ALU = store else SRW = RW.

Table 5-1 instruction types

Type	Operation	Cycles
JMPF	Goto label	5+2RW
DSI6	DS=I6	2+RW
JMPR	Goto MR	4+RW
CALL	CALL label	9+2RW+2SW
FIR_MOV	FIR_MOV_ON/OFF	2+RW
Fraction	Fraction ON/OFF	2+RW
INT SET	INT FIQ/IRQ	2+RW
IRQ	IRQ ON/OFF	2+RW
SECBANK	SECBANK ON/OFF	2+RW
FIQ	FIQ ON/OFF	2+RW
IRQ Nest Mode	IRQNEST ON/OFF	2+RW
BREAK	BREAK	10+2RW+2SW
CALLR	CALL MR	8+RW+2SW
DIVS	DIVS MR,R2	2+RW
DIVQ	DIVQ MR,R2	3+RW
EXP	R2= EXP R4	2+RW
NOP	NOP	2+RW
DS Access	DS=Rs/ Rs=Ds	2+RW
FR Access	FR=Rs/ Rs=FR	2+RW
MUL	MR = Rd* Rs,{ss,us,uu}	12+RW / 13+RW (uu)
MULS	MR = [Rd]*[Rs], size,{ss,us,uu}	us,ss : 10*N+6 + (N+1)*2*RW + {N*SW} / uu: 11*N+6 + (N+1)*2*RW + {N*SW}
Register BITOP	BITOP Rd,Rs	4+RW
Register BITOP	BITOP Rd,offset	4+RW
Memory BITOP	BITOP DS: [Rd],offset	7+2RW+SW
Memory BITOP	BITOP DS: [Rd],Rs	7+2RW+SW
Shift	Rd=Rd LSFT Rs	8+RW
RETI	RETI	8+3RW / 10+4RW (IRQ NEST ON)
RETF	RETF	8+3RW
Base+Disp6	Rd = Rd op [BP+IM6]	6+2RW
Imm6	Rd = Rd op IM6	2+RW

Type	Operation	Cycles
Branch	Jxx label	2+RW / 4+RW (taken)
Indirect	Push/Pop Rx,Ry to [Rs]	4+ 2N + (N+1)RW
DS_Indirect	Rd = Rd op DS: [Rs++]	6+RW+SRW / 7+RW+SRW (PC)
Imm16	Rd = Rs op IMM16	4+2RW / 5+2RW (PC)
Direct16	Rd = Rs op A16	7+2RW+SRW / 8+2RW+SRW (PC)
Direct6	Rd = Rd op A6	5+RW+SRW / 6+RW+SRW (PC)
Register	Rd = Rd op Rs SFT sfc	3+RW / 5+RW (PC)

- (a) MULS Cycle: $10*N+6 + (N+1)*2*RW + \{N*SW\}$ (signedxsigned, unsignedxsigned)
 $11*N+6 + (N+1)*2*RW + \{N*SW\}$ (unsignedxunsigned) where N=1..16, (N*SW) = 0 if FIR_MOVE OFF
- (b) DS_Indirect Cycle: 6 + RW + SRW / 7 + RW + SRW (write to PC)
- (c) Direct16 Cycle: 7 + 2*RW + SRW / 8 + 2*RW + SRW (write to PC)
- (d) Direct6 Cycle: 5+ RW +SRW / 6+ RW +SRW (write to PC)
- (e) RW: Memory Read Waiting cycle, RW= 0 ~ N, SW: Memory Write Waiting Cycle, SW= 0 ~N.
 SRW: store or read waiting cycle, SRW = SW when ALU = store else SRW = RW.
- (f) D: 0 (forward jump) / 1 (backward jump)
 W: 0 (not store) / 1 (store)
 DS: 0 (not using DS) / 1 (using DS)

5 Memory Organization

5.1 Introduction

The memory of SPMC75X family MCU can be separated by three blocks: SRAM, I/O port registers, and the flash. The SRAM is used for stack, variable or data storage. The I/O port register is used to control the peripheral modules. The embedded flash is designed for programming code. In addition, the SPMC75X family MCU has a 16-bit program counter, capable of addressing 64K x 16-bit. The width of address bus is assigned in A[21:0] so that the SPMC75X family MCU has the ability to address 4M x 16-bit memory. The memory allocation is shown in the Fig 6-1.

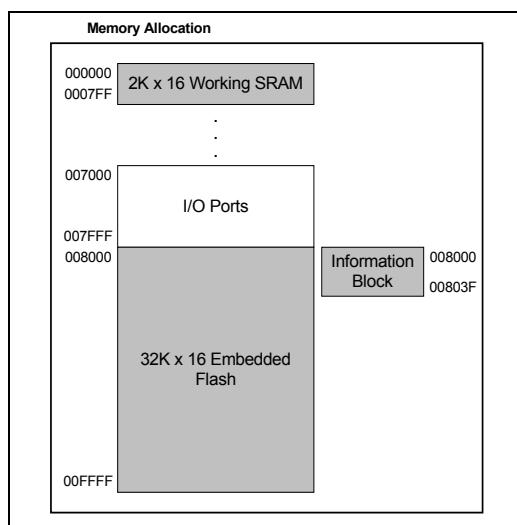


Fig 6-1 memory allocation

Note: The address of 000800 – 006FFF and 010000 – 3FFFFFF are reserved.

Table 6-1 Detail Address Mapping

I/O Address (Hex)	Mapping
0000~07FF	2KW SRAM
0800~6FFF	Illegal
7000~701F	System Control
7020~704F	Memory Control
7050~705F	Reserved
7060~709F	I/O Port Control
70A0~70AF	Interrupt Control
70B0~70BF	Time Base Control
70C0~70DF	Timer Control
70E0~70FF	Reserved

I/O Address (Hex)	Mapping
7100~711F	UART Control
7120~713F	Reserved
7140~715F	SPI Control
7160~73FF	Reserved
7400~747F	Timer/PWM Module Control (for motor control)
7500~751F	Compare Match Timer Control
7600~7FFF	Illegal
8000~FFFF	32KW Program ROM
10000~3FFFF	Illegal

5.2 Flash

The SPMC75X family MCU has two blocks of flash: the information block and normal block. Only one of the two blocks can be addressed at the same time. The information block contains 64 words. The address of information block is mapped from 0x8000 ~ 0x803F. The 0x8000 is a system option register: P_System_Option. The other address is used for user by storing important information such as version control, date, vendor name, project name, etc. The information block only can be modified by ICE or writer.

The 32K-word embedded flash is partitioned into 16 banks, 2K words each. Except one bank between 00F000 and 00F7FF can be programmed to be read-only or read-write in free run mode independently. The others 2K-word banks are read-only bank. Moreover, the 2K-word bank which can be programmed to be read-write mode also can be separated by eight frames so that the 32K embedded flash can be divided to 128 frames. The user can erase each frame separately.

5.3 SRAM

The SRAM in SPMC75X family MCU can be used for stack, variable and data storage. Stack is used for storing function call return address and pushing instruction data. The direction of stack goes from bottom to up. This stack is a FILO (first in last out) structure, and the stack address is indicated by stack pointer (SP).

The variable and data storage is configured by the user. Users can use direct access, indirect access or base pointer (BP) to load or save SRAM data. Please note that the stack and variable or storage data must not overlap each other; otherwise, program will run into an unknown status. The SPMC75X family MCU addresses maximum 2K-word SRAM. The address range is from 0x0000 to 0x07FF. In addition, the stack pointer (SP) is allocated at the end of maximum address initially.

5.4 I/O port register

The I/O port registers are used by MCU and peripheral modules for performing the desired operation of device. The I/O port registers do not exist in all of the SPMC75X family MCU. User must check the individual specification of IC body when using I/O port register. The table 6-2 shows all I/O port register address and its alias name.

Table 6-2 I/O port register address

IO Port Name	Address	IO Port Name	Address	IO Port Name	Address	IO Port Name	Address
Reserved	0x7000	Reserved	0x7010	P_IOA_Data	0x7060	P_IOC_Data	0x7070
	0x7001		0x7011	P_IOA_Buffer	0x7061	P_IOC_Buffer	0x7071
	0x7002		0x7012	P_IOA_Dir	0x7062	P_IOC_Dir	0x7072
	0x7003		0x7013	P_IOA_Attrib	0x7063	P_IOC_Attrib	0x7073
	0x7004		0x7014	P_IOA_Latch	0x7064	Reserved	0x7074
	0x7005		0x7015	Reserved	0x7065		0x7075
P_Reset_Status	0x7006		0x7016		0x7066		0x7076
P_Clk_Ctrl	0x7007		0x7017		0x7067		0x7077
Reserved	0x7008		0x7018	P_IOB_Data	0x7068	P_IOD_Data	0x7078
	0x7009		0x7019	P_IOB_Buffer	0x7069	P_IOD_Buffer	0x7079
P_WatchDog_Ctrl	0x700A		0x701A	P_IOB_Dir	0x706A	P_IOD_Dir	0x707A
P_WatchDog_Clr	0x700B		0x701B	P_IOB_Attrib	0x706B	P_IOD_Attrib	0x707B
P_Wait_Enter	0x700C		0x701C	Reserved	0x706C	Reserved	0x707C
Reserved	0x700D		0x701D		0x706D		0x707D
P_Stdby_Enter	0x700E		0x701E		0x706E		0x707E
Reserved	0x700F		0x701F		0x706F		0x707F

IO Port Name	Address	IO Port Name	Address	IO Port Name	Address	IO Port Name	Address
P_IOA_SPE	0x7080	Reserved	0x7090	P_INT_Status	0x70A0	Reserved	0x70B0
P_IOB_SPE	0x7081		0x7091	Reserved	0x70A1		0x70B1
P_IOC_SPE	0x7082		0x7092		0x70A2		0x70B2
Reserved	0x7083		0x7093		0x70A3		0x70B3
P_IOA_KCER	0x7084		0x7094	P_INT_Priority	0x70A4		0x70B4
Reserved	0x7085		0x7095	Reserved	0x70A5		0x70B5
	0x7086		0x7096		0x70A6		0x70B6
	0x7087		0x7097		0x70A7		0x70B7
	0x7088		0x7098	P_MisINT_Ctrl	0x70A8	P_TMB_Reset	0x70B8
	0x7089		0x7099	Reserved	0x70A9	P_BZO_Ctrl	0x70B9
0x708A	0x709A		0x709A		0x70AA	Reserved	0x70BA
0x708B	0x709B		0x709B		0x70AB		0x70BB
0x708C	0x709C		0x709C		0x70AC		0x70BC
0x708D	0x709D		0x709D		0x70AD		0x70BD
0x708E	0x709E		0x709E		0x70AE		0x70BE

IO Port Name	Address	IO Port Name	Address	IO Port Name	Address	IO Port Name	Address
	0x708F		0x709F		0x70AF		0x70BF
P_UART_Data	0x7100	P_SPI_Ctrl	0x7140	P_ADC_Setup	0x7160		0x7170
P_UART_RxStatus	0x7101	P_SPI_TxStatus	0x7141	P_ADC_Ctrl	0x7161		0x7171
P_UART_Ctrl	0x7102	P_SPI_TxBuf	0x7142	P_ADCData	0x7162		0x7172
P_UART_BaudRate	0x7103	P_SPI_RxStatus	0x7143		0x7163		0x7173
P_UART_Status	0x7104	P_SPI_RxBuf	0x7144	Reserved	0x7164		0x7174
Reserved	0x7105		0x7145		0x7165		0x7175
	0x7106		0x7146		P_ADC_Channel	0x7166	0x7176
	0x7107		0x7147			0x7167	0x7177
	0x7108		0x7148			0x7168	0x7178
	0x7109		0x7149			0x7169	0x7179
	0x710A		0x714A			0x716A	0x717A
	0x710B		0x714B			0x716B	0x717B
	0x710C		0x714C			0x716C	0x717C
	0x710D		0x714D			0x716D	0x717D
	0x710E		0x714E			0x716E	0x717E
	0x710F		0x714F			0x716F	0x717F

IO Port Name	Address	IO Port Name	Address	IO Port Name	Address	IO Port Name	Address
P_TMR0_Ctrl	0x7400	P_TMR0_IOCtrl	0x7410	P_TMR0_INT	0x7420	P_TMR0_TCNT	0x7430
P_TMR1_Ctrl	0x7401	P_TMR1_IOCtrl	0x7411	P_TMR1_INT	0x7421	P_TMR1_TCNT	0x7431
P_TMR2_Ctrl	0x7402	P_TMR2_IOCtrl	0x7412	P_TMR2_INT	0x7422	P_TMR2_TCNT	0x7432
P_TMR3_Ctrl	0x7403	P_TMR3_IOCtrl	0x7413	P_TMR3_INT	0x7423	P_TMR3_TCNT	0x7433
P_TMR4_Ctrl	0x7404	P_TMR4_IOCtrl	0x7414	P_TMR4_INT	0x7424	P_TMR4_TCNT	0x7434
P_TMR_Start	0x7405		0x7415	P_TMR0_Status	0x7425	P_TMR0_TPR	0x7435
P_TMR_Output	0x7406		0x7416	P_TMR1_Status	0x7426	P_TMR1_TPR	0x7436
P_TMR3_OutputCtrl	0x7407		0x7417	P_TMR2_Status	0x7427	P_TMR2_TPR	0x7437
P_TMR4_OutputCtrl	0x7408		0x7418	P_TMR3_Status	0x7428	P_TMR3_TPR	0x7438
P TPMW_Write	0x7409		0x7419	P_TMR4_Status	0x7429	P_TMR4_TPR	0x7439
P_TMR_LDOK	0x740A		0x741A		0x742A		0x743A
Reserved	0x740B		0x741B		0x742B		0x743B
	0x740C		0x741C		0x742C		0x743C
	0x740D		0x741D		0x742D		0x743D
	0x740E		0x741E		0x742E		0x743E
	0x740F		0x741F		0x742F		0x743F
P_TMR0_TGRA	0x7440	P_TMR0_TBRA	0x7450	P_TMR3_DeadTime	0x7460	P_CMT_Start	0x7500
P_TMR0_TGRB	0x7441	P_TMR0_TBRB	0x7451	P_TMR4_DeadTime	0x7461	P_CMT_Ctrl	0x7501
P_TMR0_TGRC	0x7442	P_TMR0_TBRC	0x7452	P_POS0_DectCtrl	0x7462	Reserved	0x7502
P_TMR1_TGRA	0x7443	P_TMR1_TBRA	0x7453	P_POS1_DectCtrl	0x7463		0x7503
P_TMR1_TGRB	0x7444	P_TMR1_TBRB	0x7454	P_POS0_DectData	0x7464		0x7504
P_TMR1_TGRC	0x7445	P_TMR1_TBRC	0x7455	P_POS1_DectData	0x7465		0x7505

IO Port Name	Address	IO Port Name	Address	IO Port Name	Address	IO Port Name	Address
P_TMR2_TGRA	0x7446	P_TMR2_TBRA	0x7456	P_Fault1_Ctrl	0x7466		0x7506
P_TMR2_TGRB	0x7447	P_TMR2_TBRB	0x7457	P_Fault2_Ctrl	0x7467		0x7507
P_TMR3_TGRA	0x7448	P_TMR3_TBRA	0x7458	P_OL1_Ctrl	0x7468	P_CMT0_TCNT	0x7508
P_TMR3_TGRB	0x7449	P_TMR3_TBRB	0x7459	P_OL2_Ctrl	0x7469	P_CMT1_TCNT	0x7509
P_TMR3_TGRC	0x744A	P_TMR3_TBRC	0x745A	P_Fault1_Release	0x746A		0x750A
P_TMR3_TGRD	0x744B	Reserved	0x745B	P_Fault2_Release	0x746B		0x750B
P_TMR4_TGRA	0x744C	P_TMR4_TBRA	0x745C		0x746C		0x750C
P_TMR4_TGRB	0x744D	P_TMR4_TBRB	0x745D		0x746D		0x750D
P_TMR4_TGRC	0x744E	P_TMR4_TBRC	0x745E		0x746E		0x750E
P_TMR4_TGRD	0x744F	Reserved	0x745F		0x746F		0x750F

IO Port Name	Address	IO Port Name	Address
P_CMT0_TPR	0x7510		0x7550
P_CMT1_TPR	0x7511		0x7551
	0x7512	Reserved	0x7552
	0x7513		0x7553
	0x7514		0x7554
	0x7515		P_Flash_Cmd 0x7555
Reserved	0x7516		0x7556
	0x7517		0x7557
	0x7518		0x7558
	0x7519		0x7559
	0x751A		0x755A
	0x751B		0x755B
	0x751C		0x755C
	0x751D		0x755D
	0x751E		0x755E
	0x751F		0x755F

5.5 Reset and Interrupt Vector

A reset forces the program counter (PC) points to address 0xFFFF7. When a device reset occurs, the program execution will branch to 0xFFFF7, named “Reset Vector Address”. The SPMC75X family MCU has 10 interrupts vectors. The address and function name list are given in the following table 6-3.

Table 6-3 interrupts vectors list

Reset or IRQ vector	Address
BREAK	0xFFFF5
FIQ	0xFFFF6
Reset	0xFFFF7
IRQ0	0xFFFF8
IRQ1	0xFFFF9
IRQ2	0xFFFFA
IRQ3	0xFFFFB
IRQ4	0xFFFFC
IRQ5	0xFFFFD
IRQ6	0xFFFFE
IRQ7	0xFFFFF

6 Flash Organization and Control

6.1 Introduction

The SPMC75X family MCU has two flash blocks: information block and normal block. Only one of the two blocks can be addressed at the same time. The information block contains 64 words. The address of information block is mapped from 0x8000 ~ 0x803F. The 0x8000 is a system option register P_System_Option. The other addresses are used for storing important information such version control, date, vendor name, project name etc. The information block's structure is in figure 7-1 and they only can be written in ICE mode or by writer. The 32K-word embedded flash is partitioned into 16 banks, 2K words each. Except one bank between 0xF000 and 0xF7FF can be programmed to be read-only or read-write in free run mode independently. The others 2K-word banks are read-only bank. Moreover, the 2K-word bank which can be programmed to be read-write mode also can be separated by eight frames so that the 32K embedded flash can be divided to 128 frames. The user can erase each frame separately. The relation of page and frame of flash is shown in figure 7-2.

The width of address bus is assigned in A[21:0], so the SPMC75X family MCU has the ability to address 4M x 16-bit memory.

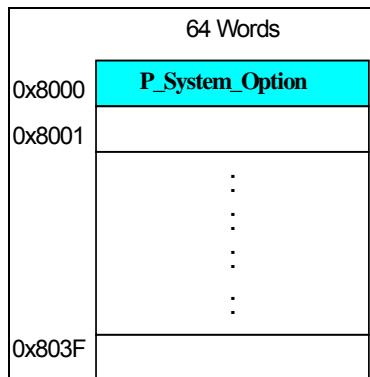


Figure 7-1 Structure of Information block

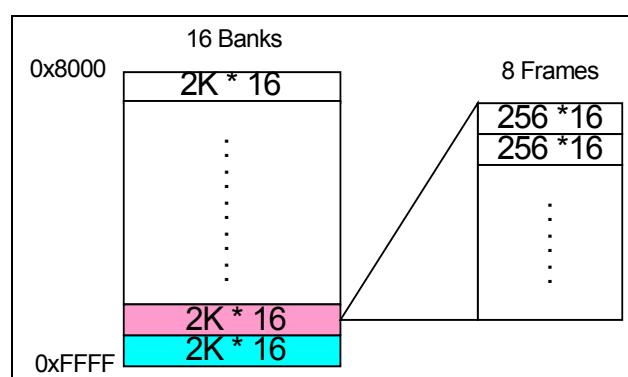


Figure 7-2 Page0 and Frame of Flash

6.2 Flash Operation

There are two registers for flash control: P_Flash_RW (0x704D) and P_Flash_Cmd (0x7555). The flash access control port, P_Flash_RW (0x704D), can be configured by two consecutive write cycles, keeping away from inadvertent writing. First, write 0x5a5a to P_Flash_RW, and then write the configuration data to P_Flash_RW within 16 clock cycles.

【Example 7-1】: Set bank14 as Read only Mode

```
#define CW_FlashRW_CMD      0x5A5A           // Flash RW Command
#define CB_BK14WDIS          (0x4000 >> 14)

P_Flash_RW->W = CW_FlashRW_CMD;           /* Flash Read Write Command */
P_Flash_RW->B.BK14WENB = CB_BK14WDIS;    /* Set Bank 14 as Read Only */
```

Listing 7-1 read-only mode for bank 14 of flash memory

The flash control register, P_Flash_Cmd, is a write only register that is for accepting/performing flash command. Before performing any commands, user should write 0xAAAA to P_Flash_Cmd for entering flash command mode at first. The Table 7-1 shows the command function and access flow.

Table 7-1 command function and access flow

	Frame Erase	Program Mode	Sequential Program Mode
1 cycle	P_Flash_Cmd = 0xAAAA		
2 cycle	[P_Flash_Cmd] = 0x5511	[P_Flash_Cmd] = 0x5533	[P_Flash_Cmd] = 0x5544
3 cycle	Set Frame Address	Write Data	Write Data
4 cycle	Wait 20ms End - Auto	Wait 40us End - Auto	Wait 40us - Auto
			Go to 2 cycle
			[P_Flash_Cmd]= 0xFFFF → Go to End

【Example 7-2】: Example for frame erasing:

```
#define CW_FlashCMD          0xAAAA           //Flash Command Flash Block
#define CW_PageErase          0x5511           //Flash Page Erase Command

unsigned int *P_WordAddr;
P_Flash_Cmd->W = CW_FlashCMD;
P_Flash_Cmd->W = CW_PageErase;
P_WordAddr = (unsigned int *)0xF000; /* P_WordAddr = start address of bank 14 */
*P_WordAddr = 0;                   /* Erase the first frame of bank 14 */
```

Listing 7-2 frame erasing of flash memory

【Example 7-3】: Example for program mode: Write 0x1234 to the address of 0xF000

```
#define CW_FlashCMD          0xAAAA           //Flash Command Flash Block
#define CW_Program             0x5533           //Flash Program Command

unsigned int *P_WordAddr;
P_Flash_Cmd->W = CW_FlashCMD;
P_Flash_Cmd->W = CW_Program;
P_WordAddr = (unsigned int *)0xF000; /* P_WordAddr = start address of bank 14 */
*(unsigned int *)P_WordAddr = 0x1234; /* program one word = 0x1234 */
```

Listing 7-3 program mode of flash memory

Note: The characteristic of flash is that the data bit can only be programmed from 1 to 0, but it is not allowed to be from 0 to 1. Therefore, if users intend to program flash, the mass erase or page erase instruction must be executed first, which erase data bit from 0 to 1.

【Example 7-4】 Example for sequential program mode: Write data to flash with sequential program mode, address is from 0xF000 to 0xF020.

```
#define CW_FlashCMD          0xAAAA          //Flash Command Flash Block
#define CW_Sequential         0x5544          //Flash Sequential Program Command
#define CW_SequentialEnd      0xFFFF          //Flash Sequential Program End Command

unsigned int *P_WordAddr;
unsigned int i,uiData=1;
P_Flash_Cmd->W = CW_FlashCMD;

for(i=0xF000;i<=0xF020;i++)
{
    P_Flash_Cmd->W = CW_Sequential;
    P_WordAddr = (unsigned int *)i;           // program address is the content of i
    *(unsigned int *)P_WordAddr = uiData;     // program uiData to P_WordAddr
    uiData++;
}/* End For Loop */
P_Flash_Cmd->W = CW_SequentialEnd;
```

Listing 7-4 sequential program mode of flash memory

6.3 Flash Registers Address Table

Table 7-2 Embedded flash and system option registers

Address	Register	Name
704Dh	P_Flash_RW	Embedded flash access control register
7555h	P_Flash_Cmd	Embedded flash access command register
8000h	P_System_Option	System option register

6.4 Control Registers

6.4.1 P_Flash_RW (0x704D): Embedded Flash Access Control Register

The flash access control port, P_Flash_RW, can be configured by two consecutive write cycle to keep away from inadvertent writing. First, write 0x5A5A to P_Flash_RW, and then write configuration data to P_Flash_RW in duration of less than 16 clock cycles.

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	1	0	0	0	0	0	0
Reserved	BK14WENB	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

This port sets up banks as Read only or Read/Write mode.

	Type(default)	Name	Frame Range	Description
B[15]	Reserved			
B[14]	R(0)	BK14WENB	Frame 112~119	F000h-F7FFh access control 1= Read-only 0= Read/write
B[13:0]	Reserved			

6.4.2 P_Flash_Cmd (0x7555): Embedded flash Access Command Register

This port is used to issue flash command. Please see the table 7-1.

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FlashCmd							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FlashCmd							

6.4.3 P_System_Option (0x8000): System Option Register

This address is allocated at information block, not in the general flash address. User must use ICE function or writer to set this port. For detailed description, please check section 6.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	1	0	1	0	1	0	1
Verification Pattern							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	1	0	1	1	1	1	1
Verification Pattern		Security	Reserved	LVR	WDG	CLKS	

Bit 15:5 **Verification Pattern**

ICE or Writer will write 01010101010 to this area

Bit 4 **Security:** security selection bit

0 = protected, the normal block in the flash cannot be accessed

1 = not protect, can be readable or write-able

Bit 3 Reserved

Bit2 **LVR:** enable low voltage reset function

0 = disable

1 = enable

Bit1 **WDG:** enable watchdog function

0 = disable

1 = enable

Bit0 **CLKS:** Clock Source Selection

0 = external clock input, connect an oscillator or clock source to XTAL2.

1 = crystal, connect a crystal device between XTAL1 and XTAL2.

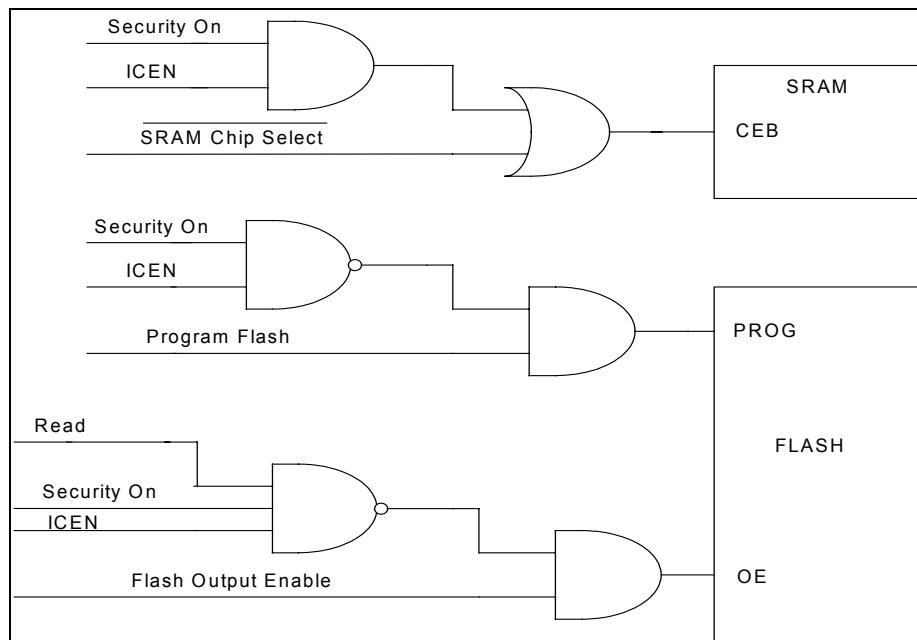
6.5 Flash Security protection

The “mass erase” command execute on main block is to erase main block only, but erase main block and information block if the command execute on information block. In case of security option in information block is activated, the SPMC75X MCU are protected from reading data through ICE or Writer function. If the security bit is turned on under ICE enable mode, the flash main block does not allow to be read/write but information block can be read by ICE and the only command that user can perform is “mass erase”. In addition, SRAM cannot be accessed (read/write) in ICE enable mode. Please refer to Table 7-1 for detail.

In normal operation (ICEN = 0), CPU can access the flash data and the working SRAM. The ICE cannot program the flash memory when the ICE mode is activated and security is turned on. This hardware protection prevents hackers from downloading a program to flash or SRAM then write source code out to GPIOs. The Figure 7-3 shows the logical block of flash security.

Table 7-1 Flash/SRAM access list in normal and ICE mode

Normal mode (ICEN=0)				
	Security =0		Security =1	
	Read	Write	Read	Write
SRAM	Yes	Yes	Yes	Yes
FLASH main block	Yes	Yes	Yes	Yes
FLASH information block	Yes	No	Yes	No
ICE mode (ICEN=1)				
SRAM	No	No	Yes	Yes
FLASH main block	No	No(but mass erase)	Yes	Yes
FLASH information block	No	No(but mass erase)	Yes	Yes


Figure 7-3 Flash security block

7 Interrupt

7.1 Introduction

The SPMC75X family MCU has up to 38 interrupt sources. These interrupt source can be grouped into two types, FIQ (Fast Interrupt Request) and IRQ0~IRQ7 (Interrupt request). Besides, the SPMC75X family MCU also implements a software interrupt, BREAK. The priority of BREAK, FIQ, IRQ is as follows: BREAK > FIQ > IRQ 0 > IRQ 1 > IRQ 2 > IRQ 3 > IRQ 4 > IRQ 5 > IRQ 6 > IRQ 7. The BREAK and FIQ is the high-priority interrupt and the IRQ is the low-priority one. On the other hand, an IRQ can be interrupted by a FIQ and BREAK, but not by another IRQ. A FIQ can only be interrupted by BREAK. If IRQNEST mode is off and more than two IRQ occurred, The priority of IRQ are IRQ0, IRQ1,IRQ2IRQ7. However, if a lower priority IRQ occurred first, even a higher priority IRQ cannot interrupt the existed IRQ. For example, if IRQ4 is occurred first, IRQ3 is unable to interrupt IRQ4. The priority takes over only when two IRQ occurred concurrently. If IRQNEST mode is on, a higher priority IRQ can interrupt the lower priority IRQ occurred first, For example, if IRQ4 is occurred first, IRQ3 is able to interrupt IRQ4. The current interrupts are listed in Table 8-1. In the table, it shows the interrupt source, interrupt name, IRQ number, and FIQ selection.

7.2 Interrupt Procedure

When the interrupt event occurred, the status is recorded and cleared only by write “1” to clear. If the event occurs and interrupt enable bit has been set, the CPU will enter interrupt request (IRQ) procedure as follows:

1. CPU jumps to interrupt vector to look up the address of corresponding interrupt service routine.
2. Push the data in SR (Status register) and the return address in PC (Program Counter register) to stack memory.
3. CPU jumps to the address of service routine to execute program and clear interrupt flag.
4. Pop the data in SR register to restore the system status. Pop the return address to PC.
5. CPU returns to the original address and keep executing the program.

The timing diagram of interrupt procedure and stack operation is as the following Figure 8-1 and Figure 8-2, respectively.

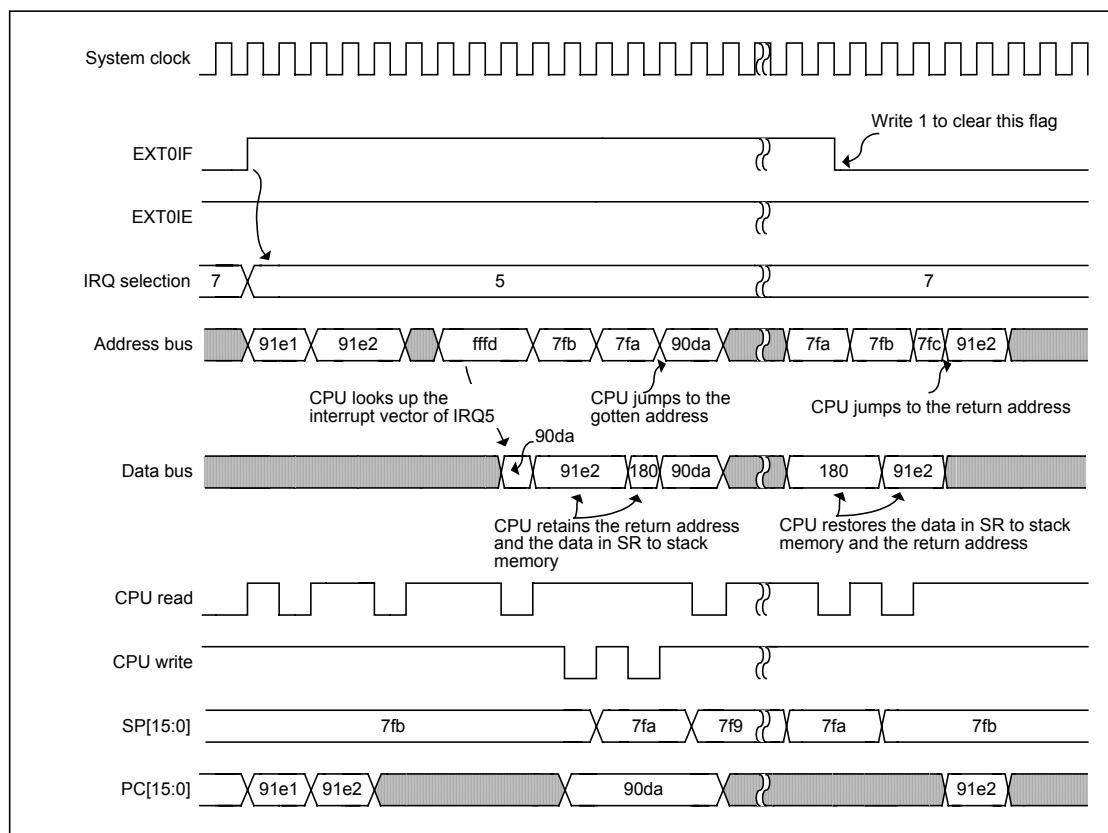


Figure 7-1 Interrupt timing

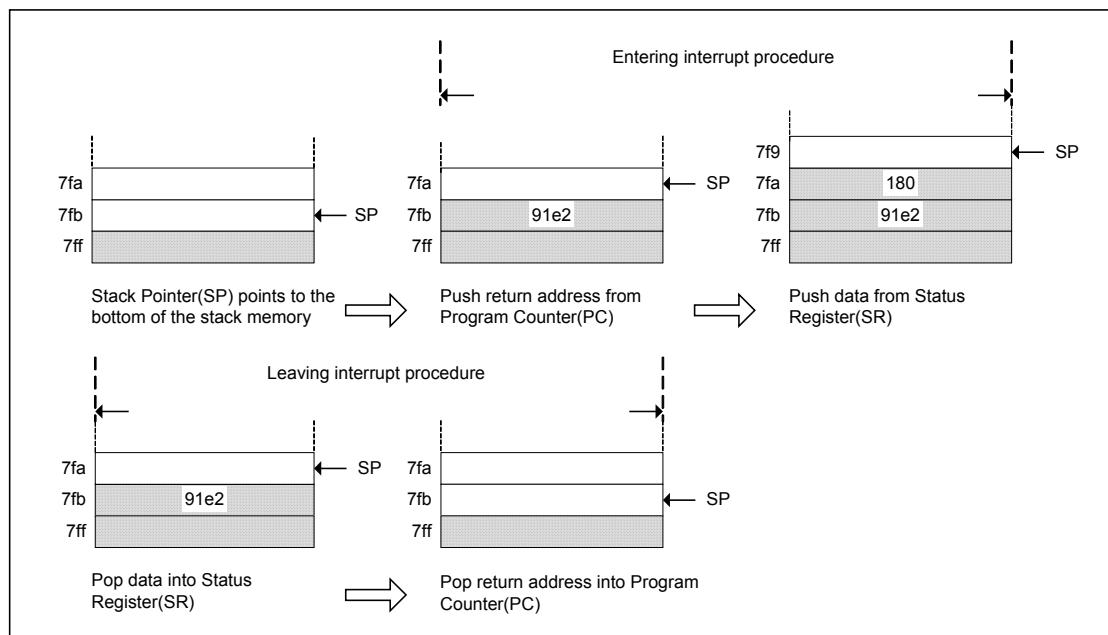


Figure 7-2 Stack memory operation with interrupt procedure

Table 8-1 Peripheral Interrupt Sharing Table

CPU IRQ	Peripheral Interrupt Source
IRQ0	Fault protection (FTINT1, FTINT2, OL1, OL2, output short) Clock monitoring (OSCFINT)
IRQ1	PDC channel 0 (TPR, TGRA, TGRB, TGRC, PDR, overflow, underflow)
IRQ2	PDC channel 1 (TPR, TGRA, TGRB, TGRC, PDR, overflow, underflow)
IRQ3	MCP channel 3 and channel 4 (TPR, TGRD)
IRQ4	TPM channel 2 (TPR, TGRA, TGRB)
IRQ5	External interrupt (EXINT0, EXINT1)
IRQ6	Serial communication interface (UART, SPI)
IRQ7	IO key change ADC conversion finished (ADCINT) Compare match timer (CMTINT0, CMTINT1)

The definitions and usages related to the interrupt configuration are listed as follows.

7.3 Interrupt Sources

The interrupt can come from the following sources (total 35 sources):

PDC Timer 0 and 1 Module

- Channel 0: TPR_0, TGRA_0, TGRB_0, TGRC_0, Position detection change, overflow, underflow
- Channel 1: TPR_1, TGRA_1, TGRB_1, TGRC_1, Position detection change, overflow, underflow

TPM Timer 2 Module

- Channel 2: TPR_2, TGRA_2, TGRB_2

MCP Timer 3 and 4 Module

- Channel 3: TPR_3, TGRD_3
- Channel 4: TPR_4, TGRD_4

Compare Match Timer

- Channel 0: CMT_0 compare match
- Channel 1: CMT_1 compare match

IO

- Key change

A/D Converter

- Conversion finished

External Interrupt

- EXINT0
- EXINT1

Serial Communication Interface

- UART
- SPI

Fault Protection

- FTINT1
- FTINT2
- OL1
- OL2
- Output short

Clock Monitoring

- Oscillator fail interrupt

Table 8-2 summarizes the interrupt source default at each IRQ level. Of course, every IRQ interrupt request can be assigned to highest level FIQ by set proper bits to '1' in P_INT_Priority register.

Table 8-2 Interrupt sources of each IRQ level

IRQ Level	Register Check Interrupt Flag	Name	Description
IRQ0 (highest)	P_INT_Status.FTIF and P_Fault1_Ctrl.FTPINIF	FTIN1_INT	Fault input pin 1 interrupt
	P_INT_Status.FTIF and P_Fault2_Ctrl.FTPINIF	FTIN2_INT	Fault input pin 2 interrupt
	P_INT_Status.FTIF and P_Fault1_Ctrl.OSF	OS1_INT	Output short 1 interrupt
	P_INT_Status.FTIF and P_Fault2_Ctrl.OSF	OS2_INT	Output short 2 interrupt
	P_INT_Status.OLIF and P_Ol1_Ctrl.OLIF	OL1_INT	Overload pin 1 interrupt
	P_INT_Status.OLIF and P_Ol2_Ctrl.OLIF	OL2_INT	Overload pin 2 interrupt
	P_INT_Status.OSCSF and P_Clk_Ctrl.OSCSF	OSCF_INT	Oscillator failed interrupt
IRQ1	P_INT_Status.PDC0IF and P_TMR0_Status.TPRIF	TPR0_INT	Timer 0 TPR interrupt
	P_INT_Status.PDC0IF and P_TMR0_Status.TGRAIF	TGRA0_INT	Timer 0 TGRA interrupt
	P_INT_Status.PDC0IF and P_TMR0_Status.TGRBIF	TGRB0_INT	Timer 0 TGRB interrupt
	P_INT_Status.PDC0IF and P_TMR0_Status.TGRCIF	TGRC0_INT	Timer 0 TGRC interrupt
	P_INT_Status.PDC0IF and P_TMR0_Status.PDCIF	PDC0_INT	Timer 0 position detection change interrupt
	P_INT_Status.PDC0IF and P_TMR0_Status.TCVIF	TCV0_INT	Timer 0 counter overflow interrupt
	P_INT_Status.PDC0IF and P_TMR0_Status.TCUIF	TUV0_INT	Timer 0 counter underflow interrupt
IRQ2	P_INT_Status.PDC1IF and P_TMR1_Status.TPRIF	TPR1_INT	Timer 1 TPR interrupt

IRQ Level	Register Check Interrupt Flag	Name	Description
	P_INT_Status.PDC1IF and P_TMR1_Status.TGAIF	TGRA1_INT	Timer 1 TGRA interrupt
	P_INT_Status.PDC1IF and P_TMR1_Status.TGBIF	TGRB1_INT	Timer 1 TGRB interrupt
	P_INT_Status.PDC1IF and P_TMR1_Status.TGCIF	TGRC1_INT	Timer 1 TGRC interrupt
	P_INT_Status.PDC1IF and P_TMR1_Status.PDCIF	PDC1_INT	Timer 1 position detection change interrupt
	P_INT_Status.PDC1IF and P_TMR1_Status.TCVIF	TCV1_INT	Timer 1 counter overflow interrupt
	P_INT_Status.PDC0IF and P_TMR1_Status.TCUIF	TUV1_INT	Timer 1 counter underflow interrupt
IRQ3	P_INT_Status.MCP3IF and P_TMR3_Status.TPRIF	TPR3_INT	Timer 3 TPR interrupt
	P_INT_Status.MCP3IF and P_TMR3_Status.TGdif	TGRD3_INT	Timer 3 TGRD interrupt
	P_INT_Status.MCP4IF and P_TMR4_Status.TPRIF	TPR4_INT	Timer 4 TPR interrupt
	P_INT_Status.MCP4IF and P_TMR4_Status.TGdif	TGRD4_INT	Timer 4 TGRD interrupt
IRQ4	P_INT_Status.TPM2IF and P_TMR2_Status.TPRIF	TPR2_INT	Timer 2 TPR interrupt
	P_INT_Status.TPM2IF and P_TMR2_Status.TGAIF	TGRA2_INT	Timer 2 TGRA interrupt
	P_INT_Status.TPM2IF and P_TMR2_Status.TGBIF	TGRB2_INT	Timer 2 TGRB interrupt
IRQ5	P_INT_Status.EXT0IF	EXT0_INT	External 0 interrupt
	P_INT_Status.EXT1IF	EXT1_INT	External 1 interrupt
IRQ6	P_INT_Status.UARTIF and P_UART_Status.RXIF	UART_RX_INT	UART receive complete interrupt
	P_INT_Status.UARTIF and P_UART_Status.TXIF	UART_TX_INT	UART transmit ready interrupt
	P_INT_Status.SPIIF and P_SPI_RxStatus.SPIRXIF	SPI_RX_INT	SPI receive interrupt
	P_INT_Status.SPIIF and P_SPI_TxStatus.SPITXIF	SPI_TX_INT	SPI transmit interrupt
IRQ7 (Lowest)	P_INT_Status.KEYIF	IOKEY_INT	IO Key change interrupt
	P_INT_Status.ADCIF and P_ADC_Ctrl.ADCIF	ADC_INT	ADC conversion complete interrupt
	P_INT_Status.CMTIF and P_CMT_Ctrl.CM0IF	CMT0_INT	Compare match timer 0 interrupt
	P_INT_Status.CMTIF and P_CMT_Ctrl.CM1IF	CMT1_INT	Compare match timer 1 interrupt

7.4 Interrupt Registers Address Table

Table8-3 Interrupt registers

Address	Register	Name
70A0h	P_INT_Status	Interrupt status register
7055h	P_INT_Priority	Interrupt priority register
70A8h	P_MisINT_Ctrl	Miscellaneous interrupt control register

7.5 Control Registers

The register passes the interrupt flags of key change, UART, SPI, EXT1, EXT0, analog-to-digital converter, MCP Timer 4, MCP Timer 3, TPM Timer 2, PDC Timer 1, PDC Timer 0, Compare Match Timer, oscillator status flag. The register is used for software polling when all interrupts are disabled. Only the KEYIF, EXT1IF and EXT0IF can write ‘1’ to clear these flags. Other are the status flags from other registers and read only.

7.5.1 P_INT_Status (0x70A0): Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
KEYIF	UARTIF	SPIIF	EXT1IF	EXT0IF	ADCIF	MCP4IF	MCP3IF

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TPM2IF	PDC1IF	PDC0IF	CMTIF	Reserved	OLIF	OSCSF	FTIF

Bit 15 **KEYIF**: Key-change interrupt status flag. This flag indicates whether a key-change interrupt has occurred.

0 = Not occurred

1 = Has occurred

Bit 14 **UARTIF**: UART interrupt status flag. This flag indicates whether a UART interrupt has occurred.

0 = Not occurred

1 = Has occurred

Bit 13 **SPIIF**: SPI interrupt status flag. This flag indicates whether a SPI interrupt has occurred.

0 = Not occurred

1 = Has occurred

Bit 12 **EXT1IF**: External interrupt 1 status flag. This flag indicates whether an external 1 interrupt has occurred..

0 = Not occurred

1 = Has occurred

- Bit 11 **EXT0IF**: External interrupt 0 status flag. This flag indicates whether an external 0 interrupt has occurred.
0 = Not occurred
1 = Has occurred
- Bit 10 **ADCIF**: A/D-converter interrupt status flag. This flag indicates whether an ADC conversion complete interrupt has occurred.
0 = Not occurred
1 = Has occurred
- Bit 9 **MCP4IF**: Timer/PWM module channel 4 interrupt status flag. This flag indicates whether a MCP channel 4 interrupt has been occurred.
0 = MCP channel 4 interrupt not occurred
1 = MCP channel 4 interrupt has occurred
- Bit 8 **MCP3IF**: Timer/PWM module channel 3 interrupt status flag. This flag indicates whether a MCP channel 3 interrupt has been occurred.
0 = MCP channel 3 interrupt not occurred
1 = MCP channel 3 interrupt has occurred
- Bit 7 **TPM2IF**: Timer/PWM module channel 2 interrupt status flag. This flag indicates whether a TPM channel 2 interrupt has been occurred.
0 = TPM channel 2 interrupt not occurred
1 = TPM channel 2 interrupt has occurred
- Bit 6 **PDC1IF**: Timer/PWM module channel 1 interrupt status flag. This flag indicates whether a PDC channel 1 interrupt has been occurred.
0 = PDC channel 1 interrupt not occurred
1 = PDC channel 1 interrupt has occurred
- Bit 5 **PDC0IF**: Timer/PWM module channel 0 interrupt status flag. This flag indicates whether a PDC channel 0 interrupt has been occurred.
0 = PDC channel 0 interrupt not occurred
1 = PDC channel 0 interrupt has occurred
- Bit 4 **CMTIF**: Compare match timer interrupt status flag. This flag indicates whether a compare match timer interrupt has been occurred.
0 = CMT interrupt not occurred
1 = CMT interrupt has occurred
- Bit 3 Reserved
- Bit 2 **OLIF**: Overload interrupt status flag. This flag indicates whether an overload condition has been occurred.
0 = Overload interrupt not occurred
1 = Overload interrupt has occurred

- Bit 1 **OSCSF**: Oscillator status flag. This flag indicates whether the on-chip crystal oscillator or external input clock runs normally. If the oscillator or external clock input stopped, this bit will be set to '1'.
- 0 = Oscillator operates normally
 1 = Oscillator failed
- Bit 0 **FTIF**: Fault protection interrupt status flag. This flag indicates if a fault protection interrupt has been occurred. There are four possible sources: FTINP1, FTINP2, TIO3A~F output short, TIO4A~F output short. If any of the above four interrupt flag has been set to '1', this flag will be read as '1'.
- 0 = Not occurred
 1 = Occurred 0:

7.5.2 P_INT_Priority (0x70A4): IRQ and FIQ Priority Selection Register

This port can set interrupt source as IRQ or FIQ. The default interrupt source is IRQ. Note: Only one of interrupt source can be set as FIQ at P_INT_Priority.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KEYIP	UARTIP	SPIIP	Reserved	EXTIP	ADCIP	MCP4IP	MCP3IP

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TPM2IP	PDC1IP	PDC0IP	CMTIP	Reserved	OLIP	OSCIP	FTIP

Bit 15 **KEYIP**: Key-change interrupt priority select bit. This bit selects the key-change interrupt priority.

- 0 = IRQ7
 1 = FIQ

Bit 14 **UARTIP**: UART interrupt priority select bit. This bit selects the UART interrupt priority.

- 0 = IRQ6
 1 = FIQ

Bit 13 **SPIIP**: SPI interrupt priority select bit. This bit selects the SPI interrupt priority.

- 0 = IRQ6
 1 = FIQ

Bit 12 Reserved

Bit 11 **EXTIP**: External interrupt priority select bit. This bit selects the external interrupt priority.

- 0 = IRQ5
 1 = FIQ

- Bit 10 **ADCIP**: ADC interrupt priority select bit. This bit selects the ADC interrupt priority.
0 = IRQ7
1 = FIQ
- Bit 9 **MCP4IP**: MCP channel 4 interrupt priority select bit. This bit selects the MCP channel 4 interrupt priority.
0 = IRQ3
1 = FIQ
- Bit 8 **MCP3IP**: MCP channel 3 interrupt priority select bit. This bit selects the MCP channel 3 interrupt priority.
0 = IRQ3
1 = FIQ
- Bit 7 **TPM2IP**: TPM channel 2 interrupt priority select bit. This bit selects the TPM channel 2 interrupt priority.
0 = IRQ4
1 = FIQ
- Bit 6 **PDC1IP**: PDC channel 1 interrupt priority select bit. This bit selects the PDC channel 1 interrupt priority.
0 = IRQ2
1 = FIQ
- Bit 5 **PDC0IP**: PDC channel 0 interrupt priority select bit. This bit selects the PDC channel 0 interrupt priority.
0 = IRQ1
1 = FIQ
- Bit 4 **CMTIP**: CMT interrupt priority select bit. This bit selects the CMT interrupt priority
0 = IRQ7
1 = FIQ
- Bit 3 Reserved
- Bit 2 **OLIP**: Overload interrupt priority select bit. This bit selects the overload interrupt priority.
0 = IRQ0
1 = FIQ
- Bit 1 **OSCIP**: Oscillator fail interrupt priority select bit. This bit selects the oscillator fail interrupt priority.
0 = IRQ0
1 = FIQ
- Bit 0 **FTIP**: Fault protection interrupt priority select bit. This bit selects the fault protection interrupt priority.
0 = IRQ0
1 = FIQ

7.5.3 P_MisINT_Ctrl (0x70A8): Miscellaneous Interrupt Control Register

This port can be set to enable interrupt. Write “1” to any bit to enable the interrupt.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
KEYIE	EXT1MS	EXT0MS	EXT1IE	EXT0IE	Reserved		

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Bit 15 **KEYIE**: Key-change interrupt enable bit

0 = Disable

1 = Enable

Bit 14 **EXT1MS**: External interrupt 1 trigger edge select bit. This bit selects the trigger edge for EXINT1.

0 = Falling edge trigger

1 = Rising edge trigger

Bit 14 **EXT0MS**: External interrupt 0 trigger edge select bit. This bit selects the trigger edge for EXINT0.

0 = Falling edge trigger

1 = Rising edge trigger

Bit 12 **EXT1IE**: External interrupt 1 enable bit. The external interrupt input EXINT1 will be sampled by Fck/2 clock. Any pulse shorter than four sample clocks will be filtered out.

0 = Disable

1 = Enable

Bit 11 **EXT0IE**: External interrupt 0 enable bit. The external interrupt input EXINT0 will be sampled by Fck/2 clock. Any pulse shorter than four sample clocks will be filtered out.

0 = Disable

1 = Enable

Bit 10:0 Reserved

7.6 Information Saving During Interrupts

During an interrupt, the returned PC value and SR registers (status register) is saved on the stack by CPU core. Typically, users may save key information registers during an interrupt e.g. R1~R5 register .This has to be implemented by software programming. The action of saving information is commonly referred to as “PUSHing,” while the action of restoring the information before the interrupt

return is commonly referred to as “POPPing.” These (PUSH, POP) are not instruction mnemonics, but are conceptual actions. Any interrupt should be returned by “RETI” instruction, or the stack will crash when stack pointer overlaps with data area.

【Example 8-1】 stores and restores the STATUS and R1~R4 registers for devices with common RAM.

```
push    r1,r5  to [sp]
.....
pop     r1,r5  from [sp]
reti
```

7.7 Interrupts prototype

The following instruction shows the prototype of the interrupt service routine in SPMC75X family MCU. The name of the interrupt entry points is reserved by the compiler. Please do not change the name of entry points.

A) Use assemble language

```
//*****
// Function: Fast Interrupt Service routine Area
// Service for (1) FIQ
//           (2) IRQ 0 ~ IRQ 7
// User's FIQ must hook on here
// _FIQ:          // Fast interrupt entrance
// _IRQ1:         // interrupt entrance
// _IRQ2:         // interrupt entrance
// _IRQ3:         // interrupt entrance
// _IRQ4:         // interrupt entrance
// _IRQ5:         // interrupt entrance
// _IRQ6:         // interrupt entrance
// _IRQ7:         // interrupt entrance
//*****
.include spmc75f_regs.inc      // include IO information
.TEXT
.public _BREAK;
.public _FIQ;
.public  _IRQ0,_IRQ1,_IRQ2,_IRQ3,_IRQ4,_IRQ5,_IRQ6,_IRQ7

//=====
//Function: Interrupt Service routine Area
```

```
//Service for FIQ
//=====
_FIQ:
    push r1,r5 to [sp];
    //-----
    //Add FIQ Function
    //-----
    pop r1,r5 from [sp];
    reti;
//=====

// Function: Interrupt Service routine Area
// Service for  IRQ1 - IRQ7
// User's IRQ must hook on here
//=====

_BREAK:
    push r1,r5 to [sp];
    //-----
    //Add BREAK Function
    //-----
    pop r1,r5 from [sp];
    reti;
//=====

_IRQ0:
    push r1,r5 to [sp];
    //-----
    // Add FTINT function
    //-----
    pop r1,r5 from [sp];
    reti;

//=====

_IRQ1:
    push r1,r5 to [sp];
    //-----
    //Add PDC0 Interrupt Function
    //-----
    pop r1,r5 from [sp];
    reti;
```

```
//=====
_IRQ2:
    push r1,r5 to [sp];
//-----
//Add PDC1 Interrupt Function

//-----
pop r1,r5 from [sp];
reti;

//=====
_IRQ3:
    push r1,r5 to [sp];
//-----
//Add MCP3 and MCP4 Interrupt Function

//-----
pop r1,r5 from [sp];
reti;

//=====
_IRQ4:
    push r1,r5 to [sp];
//-----
//Add TPM2 Interrupt Function

//-----
pop r1,r5 from [sp];
reti;

//=====
_IRQ5:
    push r1,r5 to [sp];
//-----
//Add EXINT Function
//-----
pop r1,r5 from [sp];
reti;
```

```
//=====
_IRQ6:
    push r1,r5 to [sp];
//-----
//Add UART and SPI Function

//-----
pop r1,r5 from [sp];
reti;

//=====
_IRQ7:
    push r1,r5 to [sp];
//-----
//Add KEYINT, ADCINT, CMTINT Function

//-----
pop r1,r5 from [sp];
reti;

//=====
// End of isr.asm
//=====
```

B) Use C language

```
void BREAK(void) __attribute__ ((ISR));
void BREAK(void)
{
}

void FIQ(void) __attribute__ ((ISR));
void FIQ(void)
{
}

void IRQ0(void) __attribute__ ((ISR));
void IRQ0(void)
{
}

void IRQ1(void) __attribute__ ((ISR));
void IRQ1(void)
{
}

void IRQ2(void) __attribute__ ((ISR));
void IRQ2(void)
{
}

void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
}

void IRQ4(void) __attribute__ ((ISR));
void IRQ4(void)
{
}

void IRQ5(void) __attribute__ ((ISR));
void IRQ5(void)
{
}

void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{
}
```

```
void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
}

}
```

7.8 Design Tips

When using interrupt source, the steps given below must be followed.

1. Enable Interrupt through the following control registers or from the control registers of peripherals:
P_MisINT_Ctrl, P_TMR0_INT, P_TMR1_INT, P_TMR2_INT, P_TMR3_INT, P_TMR4_INT,
P_CMT_Ctrl, P_SPI_TxStatus, P_SPI_RxStatus and P_UART_Ctrl registers.
2. Enable special function features of the specified IO port register through setting the P_IOA_SPE, P_IOB_SPE or P_IOC_SPE registers.
3. Set P_INT_Priority (0x7061) to configure IRQ and FIQ selection for interrupt
4. Use the following instructions to enable IRQ or FIQ.
 - a. IRQ ON
 - b. INT IRQ
 - c. INT FIQ
 - d. INT FIQ, IRQ

【Example 8-1】Key Change Interrupt

```
/* Set IOA[15] as key change interrupt source and enable interrupt */
    P_IOA_SPE->W      = 0x0000;          /* Disable IOA special function */
    P_IOA_Dir->W      = 0x0000;          /* Set as input */
    P_IOA_Attrib->W   = 0x0000;          /* input un-floating */
    P_IOA_Buffer->W   = 0x0000;          /* Pull High */
    P_IOA_KCER->B.KC15EN = 1;           /* Set Key change Sourge */
    P_MisINT_Ctrl->B.KEYIE = 1;          /* Enable Key change interrupt */
    Data = P_IOA_Latch->W;
    INT_IRQ();

void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
    if(P_INT_Status->B.KEYIF)           /* clear key change flag */
        P_INT_Status->B.KEYIF = 1;
}
```

Listing 8-1 key change interrupt deisng tips

【Example 8-2】 External pin Interrupt

```
/* Set IOC[2] as external interrupt source and enable interrupt.  
P_IOC_SPE->B.EXINT0EN = 1;                                /* Enable external 0 interrupt pin */  
P_MisINT_Ctrl->BEXT0IE = 1;                                /* Enable external 0 interrupt */  
INT_IRQ();  
  
void IRQ5(void) __attribute__ ((ISR));  
void IRQ5(void)  
{  
    if(P_INT_Status->BEXT0IF)  
        P_INT_Status->BEXT0IF = 1;  
}
```

Listing 8-2 external interrupt deisng tips

8 I/O Ports

8.1 Introduction

General purpose I/O can be considered the simplest peripheral. They allow the SPMC75X family MCU to monitor and control other devices. To add flexibility and functionality to a device, some parts of I/O pins are multiplexed with an alternative function. These functions can be switched through appropriate registers. For most general ports, these I/O structure contain five parts: data, buffer, direction, attribution, latch and special function enable registers. The naming rules of register are listed as follows:

Data Register	→ P_IOx_Data
Buffer Register	→ P_IOx_Buffer
Direction Register	→ P_IOx_Dir
Attribution Register	→ P_IOx_Attrib
Latch Register	→ P_IOA_Latch
Special Function Enable Register	→ P_IOA_SPE, P_IOB_SPE, P_IOC_SPE

There are four GPIOs available in this device: IOA, IOB, IOC, and IOD. Each IO is 16-bit wide. The GPIO contain some special functions in certain pins. PortA[15:8] are software programmable for wake up capability.

Each I/O pin on these 4 ports can be bit-by-bit configured by software programming. Except Port D, almost every I/O pin on these 4 ports can be programmed as special function. In other words, many special function control signals share with I/O ports. The PortA[15:8] also provide wakeup capability and key change to wakeup from power down mode .

To change PortA, PortB, and PortC from GPIO functions to special functions, all programmers need to do is enabling the corresponding special functions. This is because special functions have higher priority than GPIO does. When special functions are activated, GPIO function will be disabled.

Figure 9-1 shows the IO a sketch map. Figure 9-2 shows a typical I/O port. This does not add the multiple functions onto I/O pin. In addition, the table 9-1 is a summary of I/O setup configuration.

Table 9-1: I/O configuration

Direction	Attribution	Data	Function	Wakeup	Description
0	0	0	Pull Low	Yes**	Input with pull low
0	0	1	Pull High	Yes**	Input with pull high
0	1	0	Float*	Yes**	Input with float
0	1	1	Float	No	Input with float
1	0	0	Inverted	No	Output with data inverted (write "0" to the Data Port and will output "1" to the I/O pad)
1	0	1	Inverted	No	Output with data inverted (write "1" to the Data Port and will output "0" to the I/O pad)
1	1	0	Not Inverted	No	Output with buffer (data not inverted)
1	1	1	Not Inverted	No	Output with buffer (data not inverted)

* Default is input mode with floating state.

** Only PortA[15:8] in the state of "000", "001" and "010" have wake up capability.

All output ports contain a register P_IOx_Buffer ($x=A,B,C,D$). Therefore, the output data are retained by the register. There are two methods to write data into P_IOx_Buffer. One is P_IOx_Buffer(W), the other is P_IOx_Data(W) ($x=A,B,C,D$). P_IOx_Data(W) and P_IOx_Buffer(W) have the same result exactly. However, P_IOx_Buffer(R) reads the data stored in P_IOx_Buffer. P_IOx_Data(R) reads the input port: Port A, Port B, PortC and PortD, respectively. As a result, user should pay more attention to the operations on P_IOx_Data. For example, the data in P_IOx_Buffer and data from P_IOx_Data(R) may be different. The P_IOx_Buffer will be altered incorrectly if the bit operations SETB, CLR and INV are performed on P_IOx_Data. Therefore, it is suggested that user should perform the bit operations on P_IOx_Buffer ($x=A,B,C,D$) exclusively.

The Direction, Attribution and Data represent three ports. Each corresponding bit in these ports should be given a value. The setting rules are as follows:

- The direction setting determines whether this pin is an input or output.
- The attribute setting gives a feature to the pin, float / pull for input, not inverted/ inverted for output.
- The data setting affects the initial content of the pin. For inputs, it also determines the pull high or pull low setting. For example, suppose Port A.0 is used as input with pull low. The bit0 in Port A's Direction, Attribution and Data control ports should be given all 0s. If Port A.1 is intended to be an input with float and wakeup function, the bit1 of the Port A's Direction, Attribution and Data registers should be given a value of "010" correspondingly. Note that each port characterizes 16 Direction, Attribution and Data bits. Users should pay extra attention while configuring I/O.
- The I/O structure is able to change attribute easily. For example, the float (011) can be changed to output high (111) by only modifying the direction bit from "0" to "1".

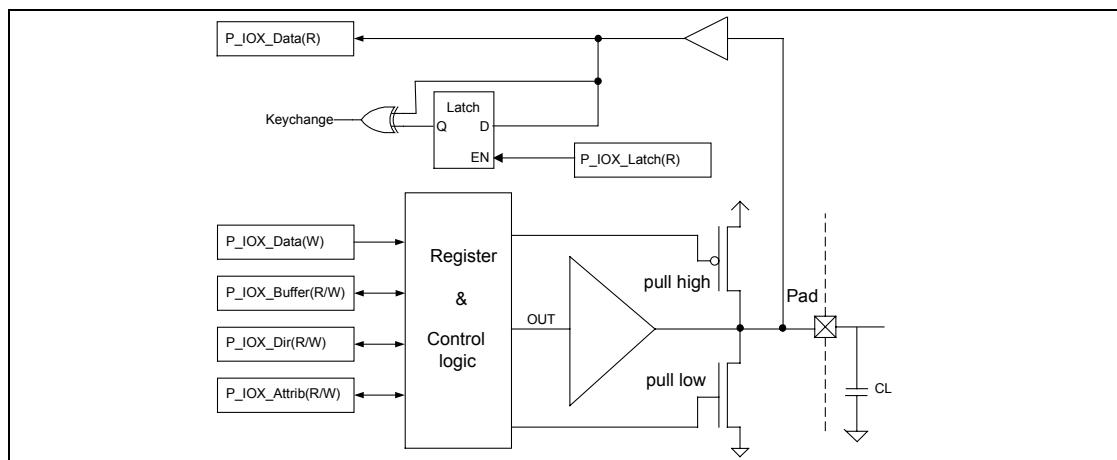


Figure 8-1 IO structure diagram

The PortA supports wakeup function. The wakeup function is available only when the I/O is configured as input pull high, low or input with float. Users can develop a low-power-consumed application by using sleep or wait function and taking the advantage of wake-up function. To reach this function, user must latch the I/O status by reading latch register. In addition, when I/O changed in sleep mode, the system will detect the I/O change and wakeup from sleep mode.

When peripheral functions are multiplexes onto general I/O pins, the functionality of I/O pins may change to accommodate the requirements of the peripheral module. Examples of this are the Analog-to-Digital(A/D) converter and SPI series interface. When the I/O pin is configured as peripheral function, the I/O direction and attribution register will be set as designate status.

All output ports contain a register such as P_IOA_Buffer, and the output data therefore are retained by the register. However, none of the input ports has a register and the input data is acquired by reading Data Register such as P_IOA_Data. Therefore, the input data is desirable to be retained externally until it is read in, or read several times before being processed. The input/output timing is shown as follows.

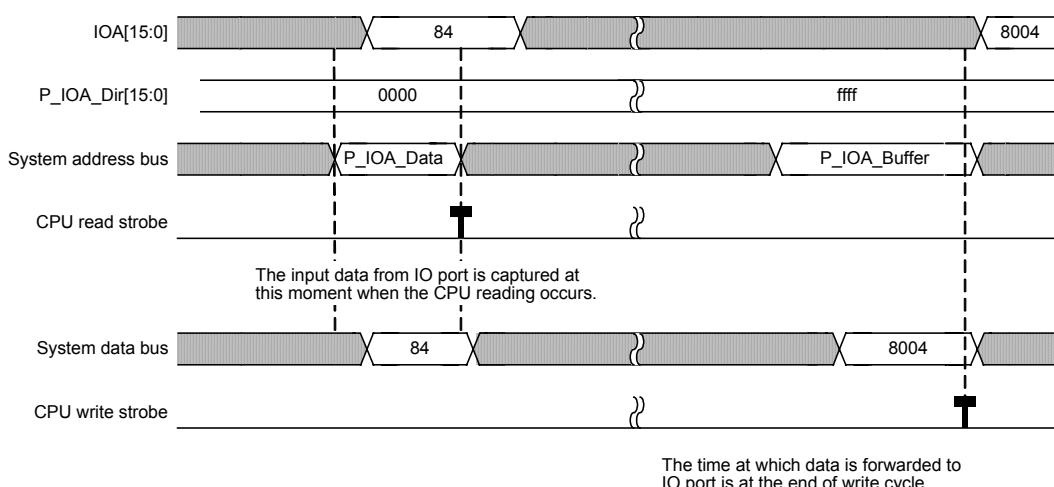


Figure 8-2 GPIO input/output timing

The open-drain configuration can be achieved by setting the registers of direction, attribution and buffer as the following table.

Table 9-2: Open-drain configuration

Direction	Attribution	Buffer	Open drain Function
0	1	0	Output high

8.1.1 Large Driving Pins

IOA[15:8], IOB[15:12], IOB[5:0], IOC[3:0], and IOC[15:10], total 28 I/O ports support large-current output capability that can direct drive LED.

8.1.2 Key-change Interrupt Pins

Key-change Wakeup is triggered if any I/O state of Port A is different from the data latched into P_IOA_Latch. The read operation for P_IOA_Latch can be performed several times to ensure that the data latched into P_IOA_Latch is correct. After the operation of “software filter”, the analog filter of IOA[15:8] is to filter out any input pulse from IO pad in comparison with the data latched into P_IOA_Latch. The Figure 9-3 shows the key change schematic block diagram and Figure 9-4 indicates the timing details.

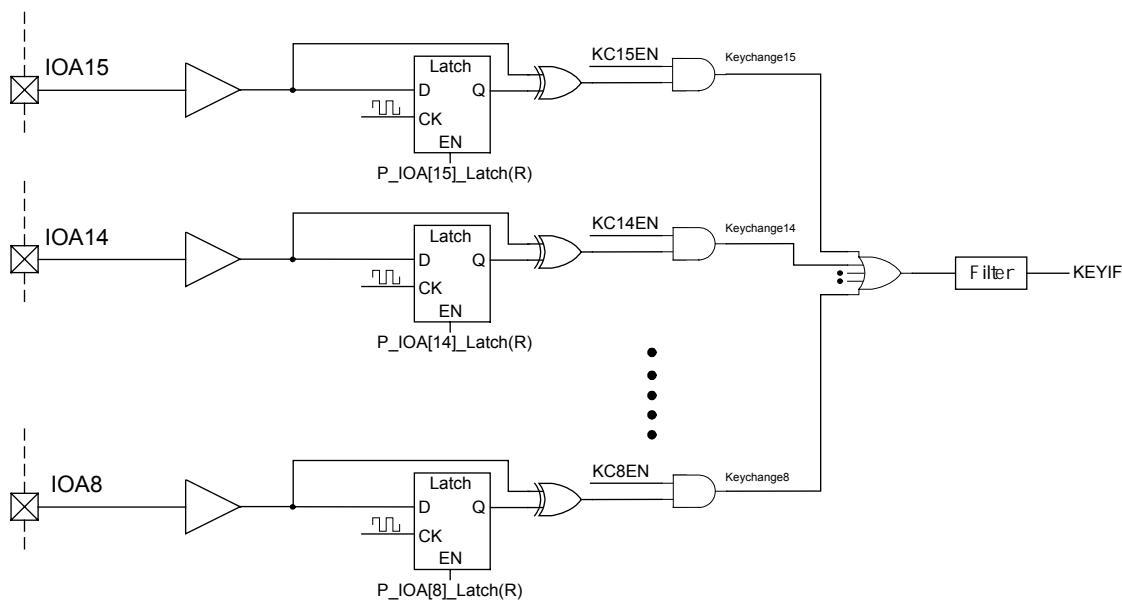


Figure 8-3 Key-change diagram

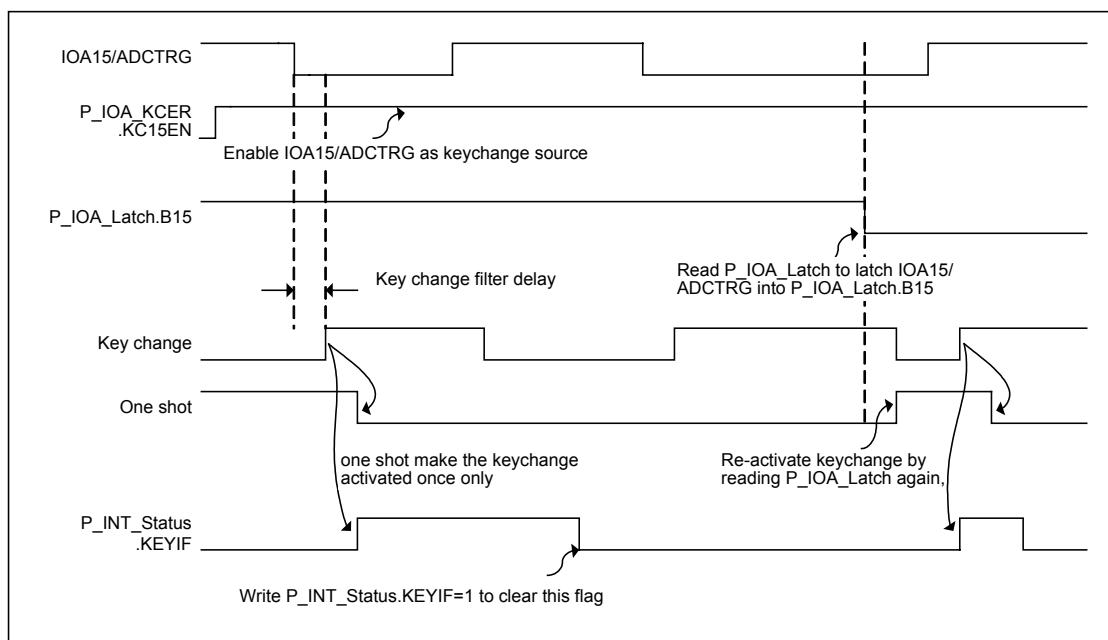


Figure 8-4 Key-change timing

8.2 IO Port Registers Address Table

Table 9-3 IO port register

Address	Register	Name
7060h	P_IOA_Data	IO port A data register
7061h	P_IOA_Buffer	IO port A buffer register
7062h	P_IOA_Dir	IO port A direction register
7063h	P_IOA_Attrib	IO port A attribute register
7064h	P_IOA_Latch	IO port A latch register
7080h	P_IOA_SPE	IO port A special function enable register
7084h	P_IOA_KCER	IO port A key change register
7068h	P_IOB_Data	IO port B data register
7069h	P_IOB_Buffer	IO port B buffer register
706Ah	P_IOB_Dir	IO port B direction register
706Bh	P_IOB_Attrib	IO port B attribute register
7081h	P_IOB_SPE	IO port B special function enable register
7070h	P_IOC_Data	IO port C data register
7071h	P_IOC_Buffer	IO port C buffer register
7072h	P_IOC_Dir	IO port C direction register
7073h	P_IOC_Attrib	IO port C attribute register
7082h	P_IOC_SPE	IO port C special function enable register
7078h	P_IOD_Data	IO port D data register
7079h	P_IOD_Buffer	IO port D buffer register
707Ah	P_IOD_Dir	IO port D direction register
707Bh	P_IOD_Attrib	IO port D attribute register

8.3 Port A

Port A contains six registers to be used for controlling the function of the general I/O. The following tables are the description of the registers.

8.3.1 P_IOA_Data (0x7060) : IO Port A Data Register

Write data into the data register and read data from the I/O pad. Writing data into P_IOA_Data will be the same as writing into P_IOA_Buffer. To prevent unwanted operation at unmodified bit data, bit operation instruction should apply at P_IOA_Buffer instead of P_IOA_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

Note: The reading of P_IOA_Data (R)(0x7060) and P_IOA_Buffer (R)(0x7061) is through different physical path. The data is from I/O pad by reading P_IOA_Data (R)(0x7060). The data is from I/O buffer by reading P_IOA_Buffer (R)(0x7061).

8.3.2 P_IOA_Buffer (0x7061) : IO Port A Buffer Register

Reading means to read data from data register. Write data into P_IOA_Buffer will be the same as writing into P_IOA_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

Note: The reading of P_IOA_Data (R)(0x7060) and P_IOA_Buffer (R)(0x7061) is through different physical path. The data is from I/O pad by reading P_IOA_Data (R)(0x7060). The data is from I/O buffer by reading P_IOA_Buffer (R)(0x7061).

8.3.3 P_IOA_Dir (0x7062) : IO Port A Direction Register

Read/Write direction-vector from/into the Direction Register.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

8.3.4 P_IOA_Attrib (0x7063) : IO Port A Attribute Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOA_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOA_Attrib							

8.3.5 P_IOA_Latch (0x7064) : IO Port A Latch Register

Read this port to latch data on the I/O PortA for key change wakeup before getting into sleep mode .

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
P_IOA_Latch							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Note: In addition, IOA[15:8] are key change wake-up sources. To activate the key change wake-up function, the P_IOA_Latch(R)(0x7004H) must be read to latch the I/O state of PortA and key change wake-up function must be enabled before entering into standby mode. Wake-up is triggered when the I/O state of PortA is different from the state at the time latched.

8.3.6 P_IOA_SPE (0x7080) : IO Port A Special Function Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	R/W	R/W	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
Reserved	TCLKDEN	TCLKCEN	TCLKBEN	TCLKAEN	TIO2BEN	TIO2AEN	Reserved

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Bit 15 Reserved

Bit 14 **TCLKDEN:** External clock D input pin(Channel 1 phase counting mode D phase input) enable

0 = Disable

1 = Enable

Bit 13 **TCLKCEN:** External clock C input pin(Channel 1 phase counting mode C phase input) enable

0 = Disable

1 = Enable

Bit 12 **TCLKBEN:** External clock B input pin(Channel 0 phase counting mode B phase input) enable

0 = Disable

1 = Enable

Bit 11 **TCLKAEN:** External clock A input pin(Channel 0 phase counting mode A phase input) enable

0 = Disable

1 = Enable

Bit 10 **TIO2BEN:** P_TMR2_TGRB input capture input/PWM output enable

0 = Disable

1 = Enable

Bit 9 **TIO2AEN:** P_TMR2_TGRA input capture input/PWM output enable

0 = Disable

1 = Enable

Bit 8:0 Reserved

8.3.7 P_IOA_KCER (0x7084) : IO Port A Key Change Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KC15EN	KC14EN	KC13EN	KC12EN	KC11EN	KC10EN	KC9EN	KC8EN

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Bit 15 **KC15EN:** PortA.15 Key change enable

0 = Disable

1 = Enable

Bit 14 **KC14EN:** PortA.14 Key change enable

0 = Disable

1 = Enable

Bit 13 **KC13EN:** PortA.13 Key change enable

0 = Disable

1 = Enable

Bit 12 **KC12EN:** PortA.12 Key change enable

0 = Disable

1 = Enable

Bit 11 **KC11EN:** PortA.11 Key change enable

0 = Disable

1 = Enable

Bit 10 **KC10EN:** PortA.10 Key change enable

0 = Disable

1 = Enable

Bit 9 **KC9EN:** PortA.9 Key change enable

0 = Disable

1 = Enable

Bit 8 **KC8EN:** PortA.8 Key change enable

0 = Disable

1 = Enable

Bit 7:0 Reserved

8.3.8 Special Functions of Port A

Port A shares with the following special functions for normal operation:

1. ADC interface → Port A [7:0] A[15]
2. External Clock input → Port A [14:11]
3. TMR2 CCP pin → Port A[10:9]

There is an enable control signal for every special function to configure designate pins as the required attribution. The control signals and description of the special function are shown in table 9-3.

Table 9-4 special function table of Port A

	SFR Pin	Type	PHB	PL	OEB	Description
IOA15	ADCETRG	I	User	User	ADCEXTRIGEN	Analog-to-digital converter external trigger to start a sample conversion
IOA14	TCLKD	I	User	User	TCLKDEN	External clock D input pin (Channel 1 phase counting mode D phase input)
IOA13	TCLKC	I	User	User	TCLKCEN	External clock C input pin (Channel 1 phase counting mode C phase input)
IOA12	TCLKB	I	User	User	TCLKBEN	External clock B input pin (Channel 0 phase counting mode B phase input)
IOA11	TCLKA	I	User	User	TCLKAEN	External clock A input pin (Channel 0 phase counting mode A phase input)
IOA10	TIO2B	User	1	0	P_IOA_Dir[10]	TGRB_2 input capture input/PWM output pin
IOA9	TIO2A	User	1	0	P_IOA_Dir[9]	TGRA_2 input capture input/PWM output pin
IOA8	-	-	-	-	-	-
IOA7	ADCCH7	I	1	0	ADCI7EN	Analog input of ADC channel 7
IOA6	ADCCH6	I	1	0	ADCI6EN	Analog input of ADC channel 6
IOA5	ADCCH5	I	1	0	ADCI5EN	Analog input of ADC channel 5
IOA4	ADCCH4	I	1	0	ADCI4EN	Analog input of ADC channel 4
IOA3	ADCCH3	I	1	0	ADCI3EN	Analog input of ADC channel 3
IOA2	ADCCH2	I	1	0	ADCI2EN	Analog input of ADC channel 2
IOA1	ADCCH1	I	1	0	ADCI1EN	Analog input of ADC channel 1
IOA0	ADCCH0	I	1	0	ADCI0EN	Analog input of ADC channel 0

8.4 Port B

Port B contains five registers to be used for controlling the function of the general I/O. The following tables are the description of the registers.

8.4.1 P_IOB_Data (0x7068) : IO Port B Data Register

Write data into data register and read from I/O pad. Writing data into P_IOB_Data will be the same as writing into P_IOB_Buffer. To prevent unwanted operation at unmodified bit data, bit operation instruction should apply at P_IOB_Buffer instead of P_IOB_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

8.4.2 P_IOB_Buffer (0x7069) : IO Port B Buffer Register

Write data into the data register and read data from the I/O buffer. Writing data into P_IOB_Buffer will be the same as writing into P_IOB_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

Note: The reading of P_IOB_Data (R)(0x7068) and P_IOB_Buffer (R)(0x7069) is through different physical path. The data is from I/O pad by reading P_IOB_Data (R)(0x7068). The data is from I/O buffer by reading P_IOB_Buffer (R)(0x7069).

8.4.3 P_IOB_Dir (0x706A) : IO Port B Direction Register

Read/Write direction vectors from/into the direction register.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

8.4.4 P_IOB_Attrib (0x706B) : IO Port B Attribute Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOB_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOB_Attrib							

8.4.5 P_IOB_SPE (0x7081) : IO Port B Special Function Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved					TIO0AEN	TIO0BEN	TIO0CEN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	1	1	1	1	1	1
OL1EN	FTIN1EN	U1EN	V1EN	W1EN	U1NEN	V1NEN	W1NEN

Bit 15:11 Reserved

Bit 10 **TIO0AEN**: P_TMR0_TGRA input capture input/PWM output pin or position detection input enable

0 = Disable

1 = Enable

- Bit 9 **TIO0BEN:** P_TMR0_TGRB input capture input/PWM output pin or position detection input enable
0 = Disable
1 = Enable
- Bit 8 **TIO0CEN:** P_TMR0_TGRC input capture input/PWM output pin or position detection input enable
0 = Disable
1 = Enable
- Bit 7 **OL1EN:** Overload protection input 1 enable
0 = Disable
1 = Enable
- Bit 6 **FTIN1EN:** External fault protection input 1 enable
0 = Disable
1 = Enable
- Bit 5 **U1EN:** U1 pin mode selection
0 = GPIO
1 = U1 phase
- Bit 4 **V1EN:** V1 pin mode selection
0 = GPIO
1 = V1 phase
- Bit 3 **W1EN:** W1 pin mode selection
0 = GPIO
1 = W1 phase
- Bit 2 **U1NEN:** U1N pin mode selection
0 = GPIO
1 = U1N phase
- Bit 1 **V1NEN:** V1N pin mode selection
0 = GPIO
1 = V1N phase
- Bit 0 **W1NEN:** W1N pin mode selection
0 = GPIO
1 = W1N phase

8.4.6 Special Function of Port B

Port B shares with the following special functions:

1. SPI interface / UART 1 Interface → Port B [13:11]
2. Capture/Compare/PWM pins → Port B [10:8]
3. TMR3 Protect → Port B [7:6]
4. TMR3 compare output → Port B [5:0]

The control signals and description of the special function pins are shown in table 9-5.

Table 9-5 special function table of Port B

	SFR Pin	Type	PHB	PL	OEB	Description
IOB[15:14]	-	-	-	-	-	-
IOB13	SDO/RXD 1	O	1	0	SPISDOEB/ UARTX1OEB	SPIEN=1, output in master and slave mode / UART transmission data output when UARTEN=1, Hi-Z when UARTRX1OEB =1
IOB12	SDI/TXD1	I	1	0	SPIEN / UARTRX1EN	Data input when SPIEN=1 / UART receive data input when UARTRX1EN =1
IOB11	SCK	I/O	1	0	SCKOEB	SPIEN=1, Input in Slave mode and Output in Master mode
IOB10	TIO0A	User	1	0	P_IOB_Dir[10]	P_Timer0_GeneralA input capture input/PWM output pin or position detection input pin
IOB9	TIO0B	User	1	0	P_IOB_Dir[9]	P_Timer0_GeneralB input capture input/PWM output pin or position detection input pin
IOB8	TIO0C	User	1	0	P_IOB_Dir[8]	P_Timer0_GeneralC input capture input/PWM output pin or position detection input pin
IOB7	OL1	I	1	0	OL1EN	Overload protection input 1
IOB6	FTINP1	I	1	0	FTIN1EN	External fault protection input pin 1
IOB5	TIO3A	O	1	0	TIO3A/U1EN TIO3HZ	U1 phase output pin
IOB4	TIO3B	O	1	0	TIO3B/V1EN TIO3HZ	V1 phase output pin
IOB3	TIO3C	O	1	0	TIO3C/W1EN TIO3HZ	W1 phase output pin
IOB2	TIO3D	O	1	0	TIO3D/U1NEN TIO3HZ	U1N phase output pin
IOB1	TIO3E	O	1	0	TIO3E/V1NEN TIO3HZ	V1N phase output pin
IOB0	TIO3F	O	1	0	TIO3F/W1NEN TIO3HZ	W1N phase output pin

8.5 Port C

Port C contains five registers to be used for controlling the function of the general I/O. The following tables are the description of the registers.

8.5.1 P_IOC_Data (0x7070) : IO Port C Data Register

Write data into data register and read from I/O pad. Writing data into P_IOC_Data will be the same as writing into P_IOC_Buffer. To prevent unwanted operation at unmodified bit data, bit operation instruction should apply at P_IOC_Buffer instead of P_IOC_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

8.5.2 P_IOC_Buffer (0x7071) : IO Port C Buffer Register

Write data into the data register and read data from the I/O buffer. Writing data into P_IOC_Buffer will be the same as writing into P_IOC_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

Note: The reading of P_IOC_Data (R)(0x7070H) and P_IOC_Buffer (R)(0x7071H) is through different physical path. The data is from I/O pad by reading P_IOC_Data (R)(0x7070H). The data is from I/O buffer by reading P_IOC_Buffer (R)(0x7071H)

8.5.3 P_IOC_Dir (0x7072) : IO Port C Direction Register

Read/Write direction vectors from/into the direction register.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

8.5.4 P_IOC_Attrib (0x7073) : IO Port C Attribute Register

Read/Write attribute vector from/into the Attribute Register.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOC_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOC_Attrib							

8.5.5 P_IOC_SPE (0x7082) : IO Port C Special Function Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	0	0
W2NEN	V2NEN	U2NEN	W2EN	V2EN	U2EN	FTIN2EN	OL2EN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TIO1CEN	TIO1BEN	TIO1AEN	Reserved	EXINT1EN	EXINT0EN	Reserved	

Bit 15 **W2NEN**: W2N pin mode selection

0 = GPIO

1 = W2N phase

Bit 14 **V2NEN**: V2N pin mode selection

- 0 = GPIO
1 = V2N phase
- Bit 13 **U2NEN:** U2N pin mode selection
0 = GPIO
1 = U2N phase
- Bit 12 **W2EN:** W2 pin mode selection
0 = GPIO
1 = W2 phase
- Bit 11 **V2EN:** V2 pin mode selection
0 = GPIO
1 = V2 phase
- Bit 10 **U2EN:** U2 pin mode selection
0 = GPIO
1 = U2 phase
- Bit 9 **FTIN2EN:** External fault protection input 2 enable
0 = Disable
1 = Enable
- Bit 8 **OL2EN:** Overload protection input 2 enable
0 = Disable
1 = Enable
- Bit 7 **TIO1CEN:** P_TMR1_TGRC input capture input/PWM output pin or position detection input enable
0 = Disable
1 = Enable
- Bit 6 **TIO1BEN:** P_TMR1_TGRB input capture input/PWM output pin or position detection input enable
0 = Disable
1 = Enable
- Bit 5 **TIO1AEN:** P_TMR1_TGRA input capture input/PWM output pin or position detection input enable
0 = Disable
1 = Enable
- Bit 4 Reserved
- Bit 3 **EXINT1EN:** External interrupt input 1 enable
0 = Disable
1 = Enable
- Bit 2 **EXINT0EN:** External interrupt input 0 enable
0 = Disable
1 = Enable
- Bit 1:0 Reserved

8.5.6 Special Function of Port C

Port C shares with the following special functions:

1. TMR4 compare output → Port C [15:10]
2. TMR4 Protect → Port C [9:8]
3. TMR1 Capture/Compare/PWM pins → Port C [7:5]
3. Buzzer output → Port C [4]
4. External Input Pin → Port C [3:2]
5. UART Pin → Port C [1:0]

The control signals and description of the special function pins are shown in table 9-6.

Table 9-6 special function table of Port C

	SFR Pin	Type	PHB	PL	OEB	Description
IOC15	TIO4F/W2N	O	1	0	TIO4F/W2NEN TIO4HZ	W2N phase output pin
IOC14	TIO4E/V2N	O	1	0	TIO4E/V2NEN TIO4HZ	V2N phase output pin
IOC13	TIO4D/U2N	O	1	0	TIO4D/U2NEN TIO4HZ	U2N phase output pin
IOC12	TIO4C/W2	O	1	0	TIO4C/W2EN TIO4HZ	W2 phase output pin
IOC11	TIO4B/V2	O	1	0	TIO4B/V2EN TIO4HZ	V2 phase output pin
IOC10	TIO4A/U2	O	1	0	TIO4A/U2EN TIO4HZ	U2 phase output pin
IOC9	FTIN2	I	1	0	FTIN2EN	External fault protection input pin 2
IOC8	OL2	I	1	0	OL2EN	Overload protection input 2
IOC7	TIO1C	User	1	0	P_IOC_Dir[7]	TGRC_1 input capture input/PWM output pin or position detection input pin
IOC6	TIO1B	User	1	0	P_IOC_Dir[6]	TGRB_1 input capture input/PWM output pin or position detection input pin
IOC5	TIO1A	User	1	0	P_IOC_Dir[5]	TGRA_1 input capture input/PWM output pin or position detection input pin
IOC4	BZO	O	1	0	BZOEBO	Buzzer output
IOC3	EXINT1	I	User	User	EXTINT1EN	External interrupt input 1
IOC2	EXINT0	I	User	User	EXTINT0EN	External interrupt input 0
IOC1	TXD2	O	1	0	UARTX2OEB	UART transmission data output when UARTEN=1, Hi-Z when UARTX2OEB =1
IOC0	RXD2	I	1	0	UARTRX2EN	UART receive data input when UARTRX2EN =1

8.6 Port D

Port D contains five registers to be used for controlling the function of the general I/O. The following tables are the description of the registers.

8.6.1 P_IOD_Data (0x7078) : IO Port D Data Register

Write data into data register and read from I/O pad. Write data into P_IOD_Data will be the same as writing into P_IOD_Buffer. To prevent unwanted operation at unmodified bit data, bit operation instruction should apply at P_IOD_Buffer instead of P_IOD_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

8.6.2 P_IOD_Buffer (0x7079) : IO Port D Buffer Register

Write data into the data register and read data from the I/O buffer. Writing data into P_IOD_Buffer will be the same as writing into P_IOD_Data.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

Note: The reading of P_IOD_Data (R)(0x7078H) and P_IOD_Buffer (R)(0x7079H) is through different physical path. The data is from I/O pad by reading P_IOD_Data (R)(0x7078H). The data is form I/O buffer by reading P_IOD_Buffer (R)(0x7079H).

8.6.3 P_IOD_Dir (0x707A) : IO Port D Direction Register

Read/Write direction vectors from/into the direction register.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

8.6.4 P_IOD_Attrib (0x707B) : IO Port D Attribute Register

Read/Write attribute vector from/into the Attribute Register.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOD_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
P_IOD_Attrib							

8.6.5 Special Function of Port D

Port D shares with the following special functions:

1. ICE Interface Pin → Port D [1:0]

The control signals and description of the special function pins are shown in table 9-7.

Table 9-7 special function table of Port D

	SFR Pin	Type	PHB	PL	OEB	Description
IOD[15:2]	-	-	-	-	-	-
IOD1	ICESDA	I/O	1	0	ICESDAOEB	ICE serial address/data input/output
IOD0	ICECLK	I	1	0	ICECLKEN	ICE serial clock input

8.7 I/O Programming Considerations

When using GPIO, design considerations need to be taken into account to ensure that the operation is as intended.

【Example 9-1】 Set IOA[3:0] as Pull Low, IOA[7:4] as Pull High, IOA[11:8] as Output Low, and IOA[15:12] as Output High.

```
P_IOA_SPE->W      = 0x0000;           /* Disable special function */
P_IOA_Dir->W      = 0xFF00;
P_IOA_Attrib->W    = 0xFF00;
P_IOA_Buffer->W    = 0xF0F0;
P_IOA_Buffer->B.Bit14 = 1;
```

Lsiting 9-1 IO Programming design tips 1

【Example 9-2】 There are two ways to read data from IOA. Reading from P_IOA_Data is to read the signal existing on the I/O pin; however, reading from P_IOA_Buffer will acquire the data on the buffer registers.

```
unsigned int Data;
Data= P_IOA_Data->W;
Data= P_IOA_Buffer->W;
```

Lsiting 9-2 IO Programming design tips 2

【Example 9-3】 There are two ways to write data to IOA. Writing P_IOA_Data is the same as writing

```
P_IOA_SPE->W      = 0x0000;           /* Disable special function */
P_IOA_Data->W      = 0xFFFF;
P_IOA_Data->B.Bit1 = 0;
P_IOA_Buffer->W    = 0x5555;
P_IOA_Buffer->B.Bit0= 0;
```

Lsiting 9-3 IO Programming design tips 3

8.8 I/O Initialization

When reset conditions occur, which are POR, LVR, External Reset and ICE Reset, the IO Port will be set in initial state, shown in table 9-7. However, when the watchdog reset and illegal reset occur, the IO port will keep the original status.

Table 9-7 Reset status

Port Name	Register Name	Address	Initial Value
Port A	P_IOA_Data	0x7060H	0000,0000,0000,0000B
	P_IOA_Buffer	0x7061H	0000,0000,0000,0000B
	P_IOA_Dir	0x7062H	0000,0000,0000,0000B
	P_IOA_Attrib	0x7063H	0000,0000,0000,0000B
	P_IOA_Latch	0x7064H	0000,0000,0000,0000B
	P_IOA_SFE	0x7080H	0000,0000,0000,0000B
	P_IOA_KCER	0x7084H	0000,0000,0000,0000B
Port B	P_IOB_Data	0x7068H	0000,0000,0000,0000B
	P_IOB_Buffer	0x7069H	0000,0000,0000,0000B
	P_IOB_Dir	0x706AH	0000,0000,0000,0000B
	P_IOB_Attrib	0x706BH	0000,0000,0000,0000B
	P_IOB_SFE	0x7082H	0000,0000,0011,1111B
Port C	P_IOC_Data	0x7070H	0000,0000,0000,0000B
	P_IOC_Buffer	0x7071H	0000,0000,0000,0000B
	P_IOC_Dir	0x7072H	0000,0000,0000,0000B
	P_IOC_Attrib	0x7073H	0000,0000,0000,0000B
	P_IOC_SFE	0x7083H	1111,11000,0000,0000B
Port D	P_IOD_Data	0x7078H	0000,0000,0000,0000B
	P_IOD_Buffer	0x7079H	0000,0000,0000,0000B
	P_IOD_Dir	0x707AH	0000,0000,0000,0000B
	P_IOD_Attrib	0x707BH	0000,0000,0000,0000B

9 Power Saving modes and wakeup

9.1 Introduction

The SPMC75X family MCU provides two modes for power saving. (Wait mode and Standby mode) The function are listed as follows:

- **Normal mode**

When device operates in normal mode, it consumes the maximum power, and all peripherals can be used.

- **Wait mode**

Only CPU is powered-down in wait mode to decrease CPU power consumption. Other peripherals keep their previous states and are operable. When wake-up, CPU will resume and execute next instruction. Figure 10-1 shows the wait mode timing.

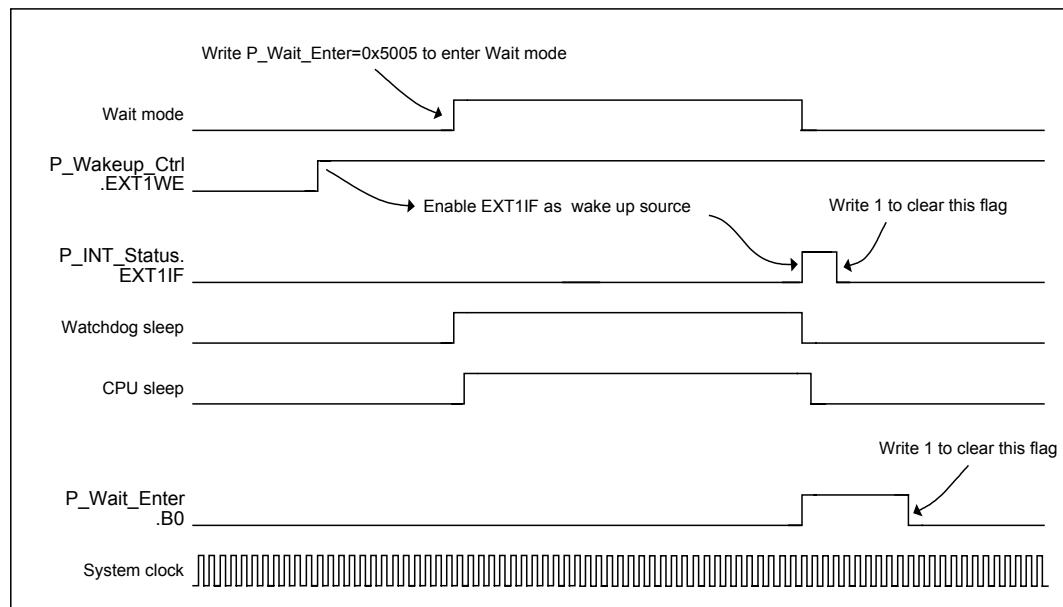


Figure 9-1 Wait mode timing waveform

- **Standby mode**

All modules are disabled in this mode. Power consumption is minimized in this mode. When wake-up, CPU will be reset and back to normal operation mode. Note that if MCP channel 3 or channel 4 (Timer3 or Timer4) has been set to PWM output mode, the device will not enter Wait mode or Standby mode. Figure 10-2 shows the standby mode timing.

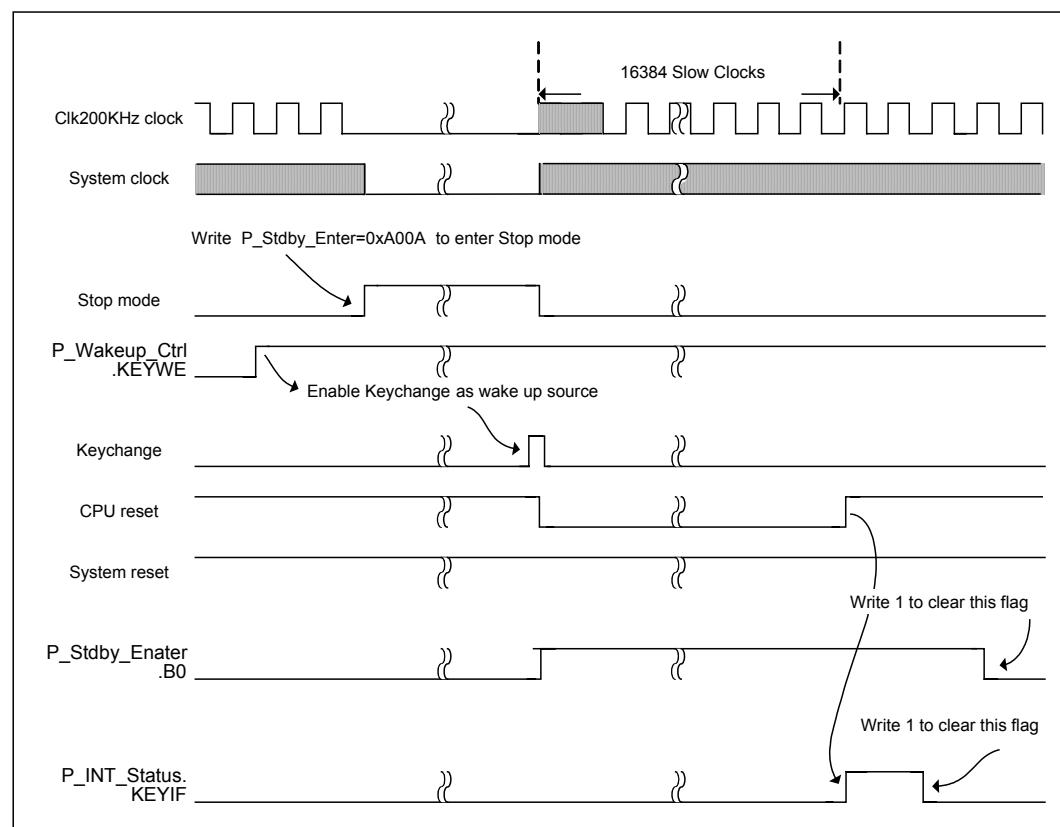


Figure 9-2 Standby mode timing

The standby mode will turn off all function and system clocks to consume the least current. All these three modes will be awoken by key-press if the key wakeup function is enabled. The Figure 10-3 shows the state transition diagram between normal mode and power saving modes.

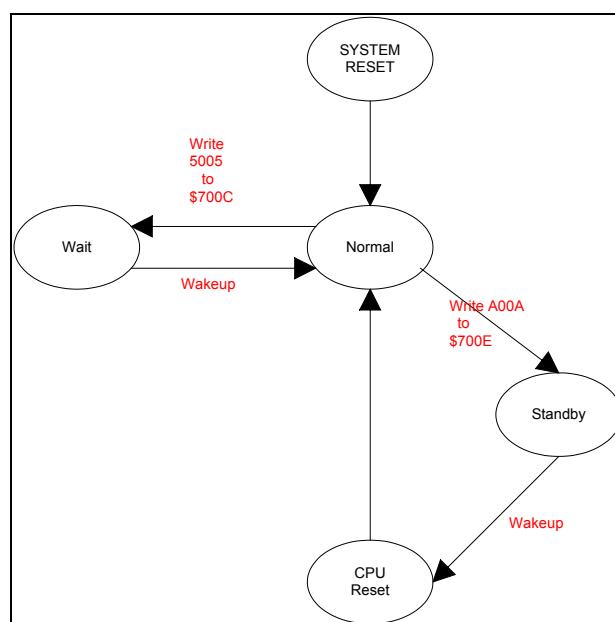


Figure 9-3 Operation modes transition state diagram

Table 10-1: the relationship of mode vs. operation

	Wait	Standby
CPU	OFF	OFF
PLL	ON	OFF
Wakeup from	Next instruction	Reset CPU

9.2 Wake-up Sources

The wake-up event may come from the following sources (total 28 sources):

- Timer Module
 - PDC Channel 0: TPR_0, TGRA_0, TGRB_0, TGRC_0, Position detection change, overflow, underflow
 - PDC Channel 1: TPR_1, TGRA_1, TGRB_1, TGRC_1, Position detection change, overflow, underflow
 - TPM Channel 2: TPR_2, TGRA_2, TGRB_2
 - MCP Channel 3: TPR_3, TGRD_3
 - MCP Channel 4: TPR_4, TGRD_4
- Compare Match Timer
 - Channel 0: CMT_0 compare match
 - Channel 1: CMT_1 compare match
- IO
 - Key change
- External Interrupt
 - EXINT0
 - EXINT1
- Serial Communication Interface
 - UART
 - SPI

9.3 Control Register

9.3.1 P_Wait_Enter (0x700C) : Wait-mode Entrance Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
WaitCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
WaitCMD							

1. Write 0x5005 to enter wait mode (CPU off, PLL on).
2. Write 0x0001 will clear wait flag.
3. Read 0x0001 indicates that it is wake-up from wait mode.
4. Note that to enter Wait mode, MCP channel 3 or 4 must not be set to PWM output mode. In ICE mode, SPMC75X family MCU body cannot enter into wait mode.

9.3.2 P_Stdby_Enter (0x700E) : Standby-mode Entrance Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
StdbyCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
StdbyCMD							

1. Write 0xA00A to enter standby mode (CPU off, PLL off).
2. Write 0x0001 will clear standby flag.
3. Read 0x0001 indicates that it is wake-up from Standby mode.
4. Note that to enter Standby mode, MCP channel 3 or 4 must not be set to PWM output mode. In ICE mode, SPMC75X family MCU body cannot enter into standby mode.

9.3.3 P_Wakeup_Ctrl (0x700F) : Wake-up Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KEYWE	UARTWE	SPIWE	EXT1WE	EXT0WE	Reserved		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TPM2WE	PDC1WE	PDC0WE	CMTWE	Reserved			

- Bit 15 **KEYWE:** Key-change wake-up enable bit
0 = Disable
1 = Enable
- Bit 14 **UARTWE:** UART wake-up enable bit
0 = Disable
1 = Enable
- Bit 13 **SPIWE:** SPI wake-up enable bit
0 = Disable
1 = Enable
- Bit 12 **EXT1WE:** External interrupt 1 wake-up enable bit
0 = Disable
1 = Enable
- Bit 11 **EXT0WE:** External interrupt 0 wake-up enable bit
0 = Disable
1 = Enable
- Bit 10:8 Reserved
- Bit 7 **TPM2WE:** TPM channel 2 wake-up enable bit
0 = Disable
1 = Enable
- Bit 6 **PDC1WE:** PDC channel 1 wake-up enable bit
0 = Disable
1 = Enable
- Bit 5 **PDC0WE:** PDC channel 0 wake-up enable bit
0 = Disable
1 = Enable
- Bit 4 **CMTWE:** Compare match timer wake-up enable bit
0 = Disable
1 = Enable
- Bit 3:0 Reserved

9.4 Design Tips

【Example 10-1】 Set MCU enter into wait mode. And clear wait mode flag

```
P_Wait_Enter->W = CW_WaitCMD;           /* Enter Wite Mode */  
P_Wait_Enter->W = CW_WaitClr;           /* Wite Mode Flag Clear */
```

Listing 10-1 MCU wait mode deisng tips

【Example 10-2】Set MCU enter into standby mode. And clear standby mode flag

```
P_Stdby_Enter->W = CW_StdbyCMD;           /* Enter Standby Mode */  
P_Stdby_Enter->W = CW_StdbyClr;           /* Standby Mode Flag Clear */
```

Listing 10-2 MCU standby mode design tips

10 PDC Timer 0 and 1 Module

10.1 Introduction

SPMC75X family MCU provides two channels of 16 bit PDC (Phase Detection Control, PDC) timers used for capture function and PWM operation. Also supports position detection features for Brushless-DC motor application. The PDC timers are very useful for mechanical speed calculation including ACI and BLDC motor. For BLDC motor, thase commutation for change current conduction is according to position information. Figure 11-1 shows the block diagram of entire PDC timers, channel 0 and channel 1. For details of PDC timers specification, please refer to Table 11-1.

10.2 PDC Timers Features

- ❑ Capability to process up to six inputs for capture or hall position input. Or six output for PWM operation.
- ❑ Six timer general registers (TGRAx/TGRBx/TGRCx, x = 0, 1): three registers for each channel independently assignable PWM or input capture functions.
- ❑ Six timer buffer registers (TBRAx/TBRBx/TBRCx, x = 0, 1) : three registers for each channel used for PWM buffering and capture operation.
- ❑ Selection of eight programmable clock source: six internal clocks (FCK/1, FCK/4, FCK/16, FCK/64, FCK/256, FCK/1024), two external clocks (TCLKA and TCLKB).
- ❑ Programmable of timer operating modes:
 1. Timer counting modes:
 - Normal counting mode: continuous up counting.
 - PWM function: selection of 1 output, 0 output at compare match and output hold.
 - Input capture function: selection of issue capture at clock rising, falling, both edge, and position detection change event.
 2. PWM mode:
 - Three independent PWM output for each channel and can be provided with desired duty ratio.
 3. Edge-aligned PWM generation mode:
 - PWM output for normal and up counting operation.
 4. Center-aligned PWM generation mode:
 - PWM output for normal and directional up-down counting operation.
 5. Position detection change (PDC) module:
 - Programmable sample clock source for position signals input: four internal clocks source (FCK/4, FCK/8, FCK32 and FCK/128).
 - Programmable position sampling mode: selection of sample position when PWM on, regularly, and lower phase are conducting current.

- Programmable sampling count for position signal to avoid glitch affects the position data.
- Totally 14 interrupt sources:
 - One timer period compare match interrupt and counter overflow/underflow interrupt sources.
 - Three TGR register compare match interrupt sources.
 - Position detection changes interrupt sources.
- Timer buffer operation:
 - The input capture register can be consisted of double buffers. The PWM timer general register can automatically be modified.
- Any initial timer compare match and period value can be set.
- Read-only 16-bit up, up/down and phase direction counter register P_TMRx_TCNT ($x = 0, 1$) register.
- R/W 16-bit timer period registers : P_TMRx_TPR ($x = 0, 1$) provides time base of system.
- Three R/W 16-bit timer general registers : P_TMRx_TGRA, P_TMRx_TGRB, P_TMRx_TGRC ($x = 0, 1$), are used for input capture and PWM output.
- Three read-only timer buffer registers : P_TMRx_TBRA, P_TMRx_TBRB, P_TMRx_TBRC ($x = 0, 1$), are the buffer of above mentioned timer general registers.
- R/W 16-bit timer control register P_TMRx_Ctrl and input output control register P_TMRx_IOCctrl ($x = 0, 1$) in which used for the selection of input capture or PWM modes.
- R/W 16-bit timer interrupt enable register P_TMRx_INT provides 7 interrupt sources, and 16-bit R/W interrupt status register P_TMRx_Status ($x = 0, 1$) in which records the interrupt flags.
- R/W 16-bit position detection control register P_POSx_DectCtrl and position detection data register P_POSx_DectData ($x = 0, 1$) for position signals feedback.

Table 11-1 PDC timers specification

Function	PDC Timer 0	PDC Timer 1
Clock sources	Internal clock FCK/1, FCK/4, FCK/16, FCK/64, FCK/256, FCK/1024	
	External clock TCLKA, TCLKB	
IO pins	◆ TIO0A ◆ TIO0B ◆ TIO0C	◆ TIO1A ◆ TIO1B ◆ TIO1C
Timer general register	◆ P_TMR0_TGRA ◆ P_TMR0_TGRB ◆ P_TMR0_TGRC	◆ P_TMR1_TGRA ◆ P_TMR1_TGRB ◆ P_TMR1_TGRC
Timer buffer Register	◆ P_TMR0_TBRA ◆ P_TMR0_TBRB ◆ P_TMR0_TBRC	◆ P_TMR1_TBRA ◆ P_TMR1_TBRB ◆ P_TMR1_TBRC
Timer period and counter register	◆ P_TMR0_TPR ◆ P_TMR0_TCNT	◆ P_TMR1_TPR ◆ P_TMR1_TCNT
Capture sample clock	Internal clock: FCK/1, FCK/2, FCK/4, FCK/8	
Counting edge	◆ Rising edge ◆ Falling edge	

Function		PDC Timer 0	PDC Timer 1
		◆ Both edge	
Counter clear source		◆ P_TMR0_TGRA, P_TMR0_TGRB, P_TMR0_TGRC capture input. ◆ P_POS0_DectData position detection data changes. ◆ P_TMR0_TPR compare matches.	◆ P_TMR1_TGRA, P_TMR1_TGRB, P_TMR1_TGRC capture input. ◆ P_POS1_DectData position detection data changes. ◆ P_TMR1_TPR compare matches.
Input capture function		Yes	Yes
PWM compare match output function	1 output	Yes	Yes
	0 output	Yes	Yes
	Output Hold	Yes	Yes
Edge-aligned PWM		Yes	Yes
Center-aligned PWM		Yes	Yes
Phase counting mode		Yes, phase inputs are TCLKA/TCLKB	Yes, phase inputs are TCLKC/TCLKD
Timer buffer operation		Yes	Yes
AD convert start trigger		P_TMR0_TGRA compare match	P_TMR1_TGRA compare match
Interrupt sources		◆ Timer 0 TPR interrupt ◆ Timer 0 TGRA interrupt ◆ Timer 0 TGRB interrupt ◆ Timer 0 TGRC interrupt ◆ Timer 0 PDC interrupt ◆ Timer 0 overflow interrupt ◆ Timer 0 underflow interrupt	◆ Timer 1 TPR interrupt ◆ Timer 1 TGRA interrupt ◆ Timer 1 TGRB interrupt ◆ Timer 1 TGRC interrupt ◆ Timer 1 PDC interrupt ◆ Timer 1 overflow interrupt ◆ Timer 1 underflow interrupt

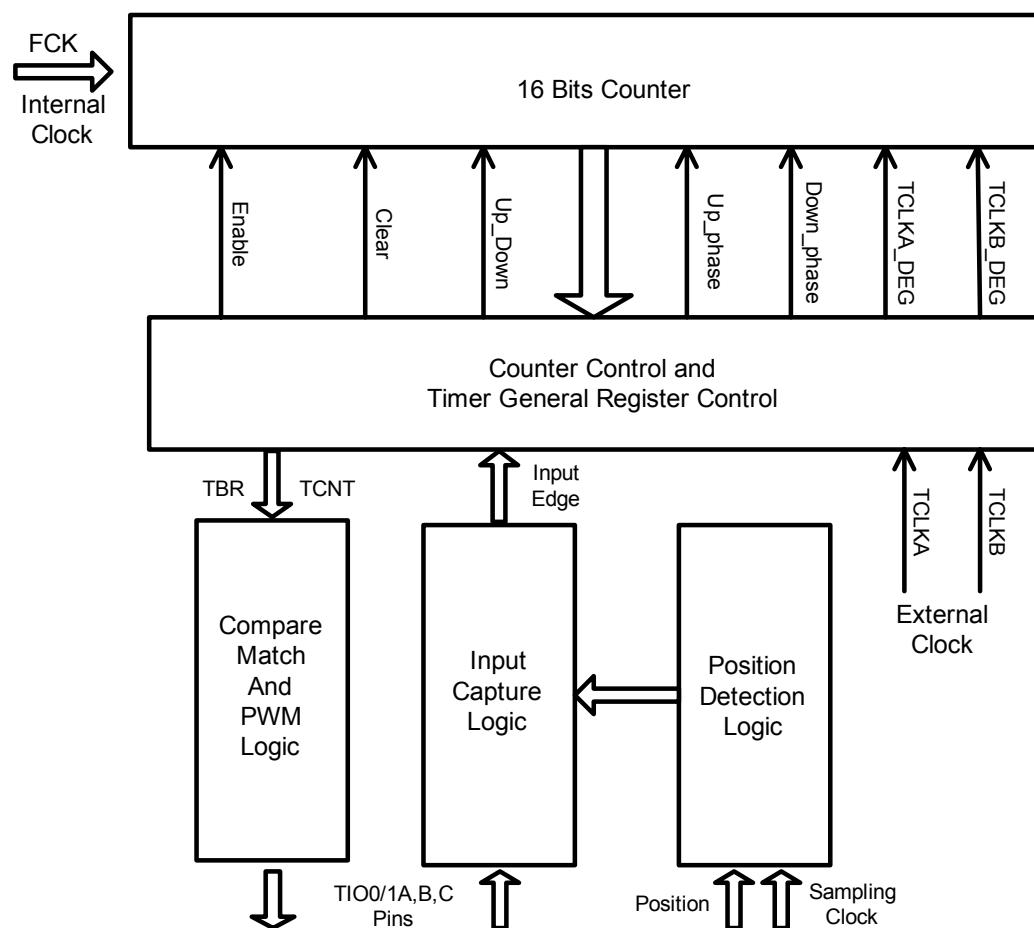


Figure 10-1 PDC timers block diagram

10.3 PDC Timers Input/Output/Special Function Pins

Channel	Pin name	I/O	Function
Common	TCLKA	Input	External clock A input pin (Channel 0 phase counting mode A phase input)
	TCLKB	Input	External clock B input pin (Channel 0 phase counting mode B phase input)
	TCLKC	Input	External clock C input pin (Channel 1 phase counting mode C phase input)
	TCLKD	Input	External clock D input pin (Channel 1 phase counting mode D phase input)
0	TIO0A	I/O	P_TMR0_TGRA input capture /PWM output pin
	TIO0B	I/O	P_TMR0_TGRB input capture/PWM output pin
	TIO0C	I/O	P_TMR0_TGRC input capture/PWM output pin
1	TIO1A	I/O	P_TMR1_TGRA input capture/PWM output pin
	TIO1B	I/O	P_TMR1_TGRB input capture/PWM output pin
	TIO1C	I/O	P_TMR1_TGRC input capture/PWM output pin

10.4 PDC Timer Counting Operation

The each on-chip PDC timer has the following five possible counting operations :

Normal operation (normal up-counting).

Directional phase counting mode 1 to 4 .

Count on external clock input pin TCLKA or TCLKB.

Edge-aligned PWM mode (continuous up counting, PWM output mode).

Center-aligned PWM mode (continuous up/down counting, PWM output mode).

The bits value MODE in the P_TMRx_Ctrl (x = 0, 1) register determines the PDC timer counting mode.

When the corresponding timer enable bit in P_TMR_Start is set to 1, the timer is starting to count from 0x0000. And the CCLS bits value in the P_TMRx_Ctrl (x = 0, 1) register determines the counter clear event.

10.4.1 Continuous Up Counting Mode with Edge-Aligned PWM

The PDC timers can be configured as edge-aligned PWM mode with PWM output or normal operation without any output waveform by setting MODE bits in P_TMRx_Ctrl (x = 0, 1) register. At this mode, the timer counter acts as up-counting timer and counts from 0x0000 to timer period register value. At this mode, user must set P_TMRx_TPR (x = 0 ~ 1) register and set counter clear source (CCLS) as cleared by timer period compare match and also needs to disable Port A/B/C specification function for compare match output pin.

The PDC timer continuous up counting according to the input clock sources from bits value TMRPS defined in corresponding timer control register. The timer counter register cleared to zero until it matches that of the timer period register and period compare match event interrupt flag TPRIF is set. The period interrupt request is generated when PPRIE bit is set in P_TMRx_INT (x = 0, 1) register.

Once the timer counter register became 0x0000, the underflow event flag TCUIF is set. The underflow interrupt request is generated when TCUIE bit is set in P_TMRx_INT (x = 0, 1) register.

The overflow flag is set when the timer counter register value reaches 0xFFFF and TCVIF flag is set. The overflow interrupt request is generated when TCVIE bit is set in P_TMRx_INT (x = 0, 1) register.

The general register compare match event occurs when timer counter register matches the content of TGRA, TGRB or TGRC register. It generates the general register compare match interrupt when TGAIE, TGBIE or TGCIE bit is set in the corresponding timer interrupt enable register.

The initial value of P_TMRx_TPR ($x = 0, 1$) can be any value from 0x0000 to 0xFFFF. Either the external clock input pin or internal clock source FCK can be selected as the clock source of the timer. The normal continuous up counting mode is extremely suitable for the generation of edge-triggered or asynchronous PWM waveforms and sampling periods in digital motor control systems. Figure 11-2 shows the normal continuous up counting mode of the PDC timer.

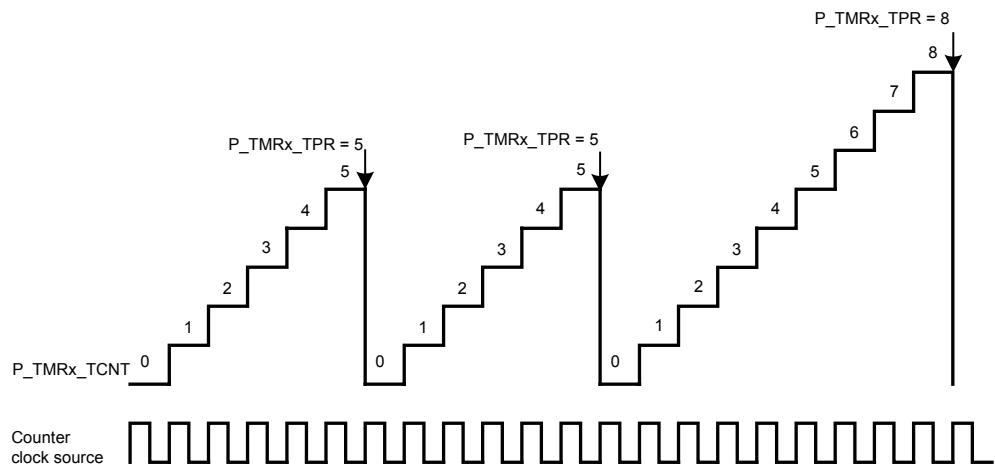


Figure 10-2 Continuous up counting example (CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

At edge-aligned PWM mode, user must set P_TMRx_TPR ($x= 0, 1$) period register and P_TMRx_TGRy ($y = A, B, C$) general register then set counter clear source (CCLS) as cleared by timer period compare match. The compare match output condition set at P_TMRx_IOCtrol ($x= 0, 1$) register. Figure 11-3 shows the normal continuous up counting mode for edge-aligned PWM generation of timer 0. And Figure 11-4 also indicates the edge-aligned PWM timing details.

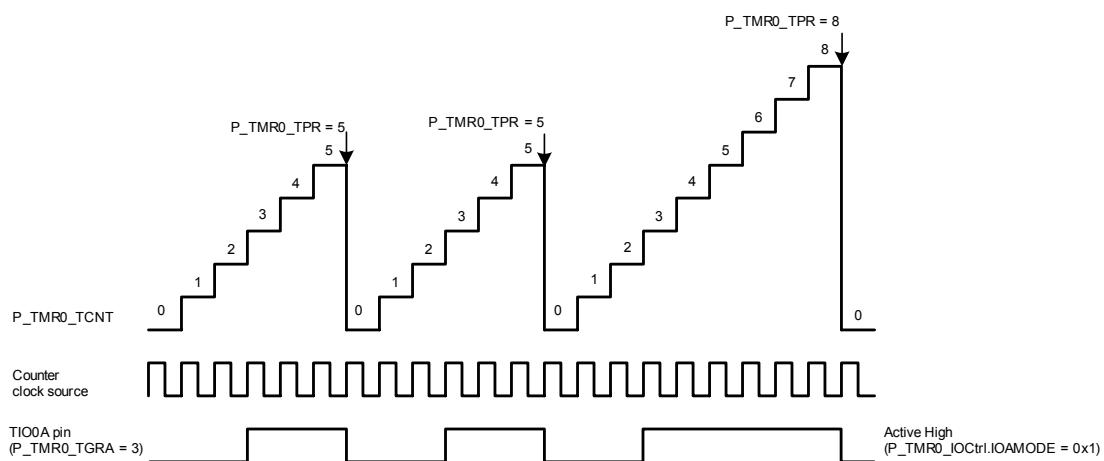


Figure 10-3 Edge-Aligned mode PWM

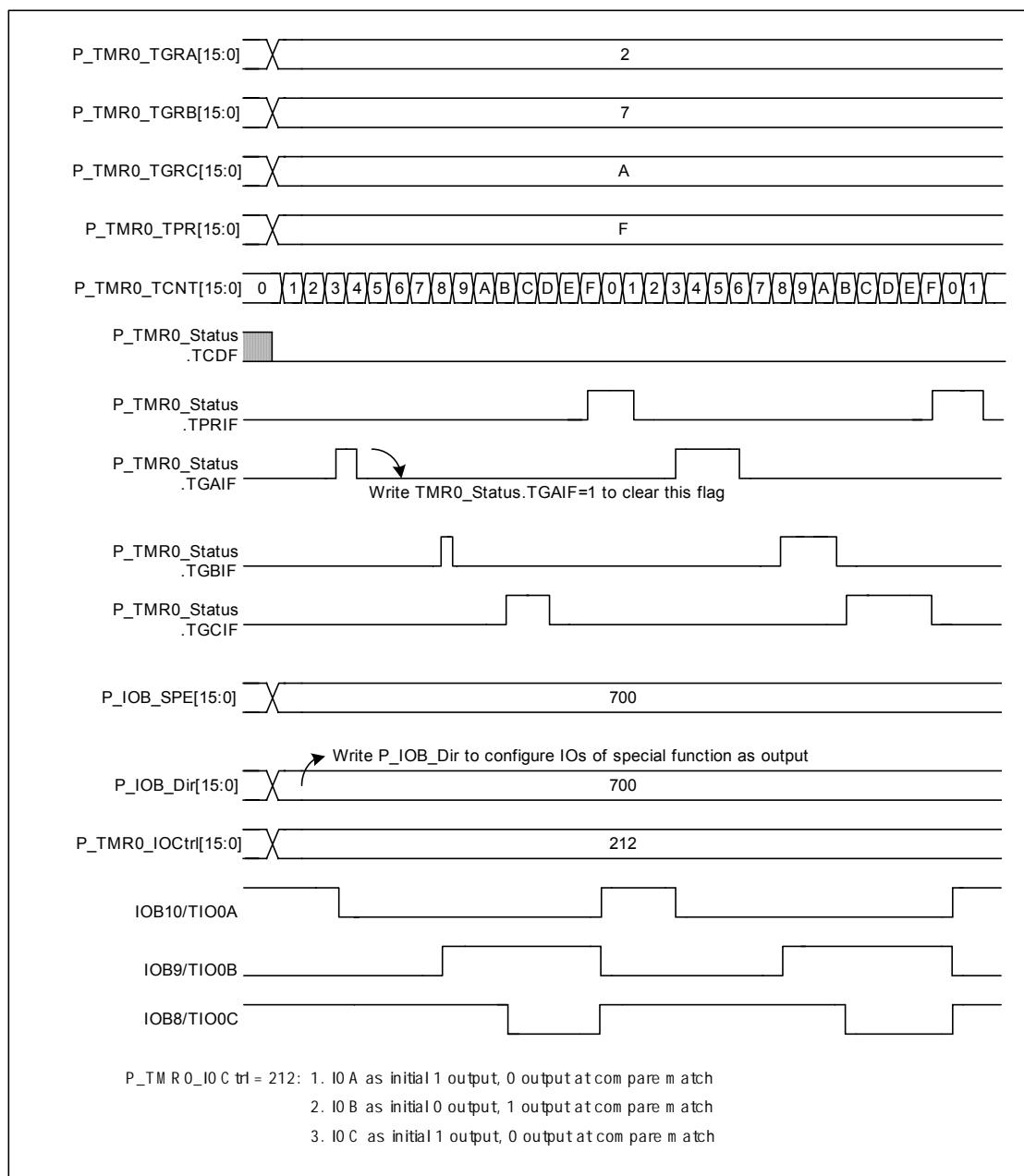


Figure 10-4 Edge-Aligned PWM timing of Timer 0

10.4.2 Continuous up/down counting mode with Center-Aligned PWM

The operation of continuous up/down counting mode is the same as up counting mode except the timer period register defines the middle transition point of whole counting process. The counting direction changes from up to down when the timer counter register reaches the timer period register. The period of the timer is two times of P_TMRx_TPR ($x = 0, 1$) of the scaled clock input and the setting of CKEGS in the P_TMRx_Ctrl ($x = 0, 1$) register. Figure 11-5 shows the continuous up/down counting mode operation.

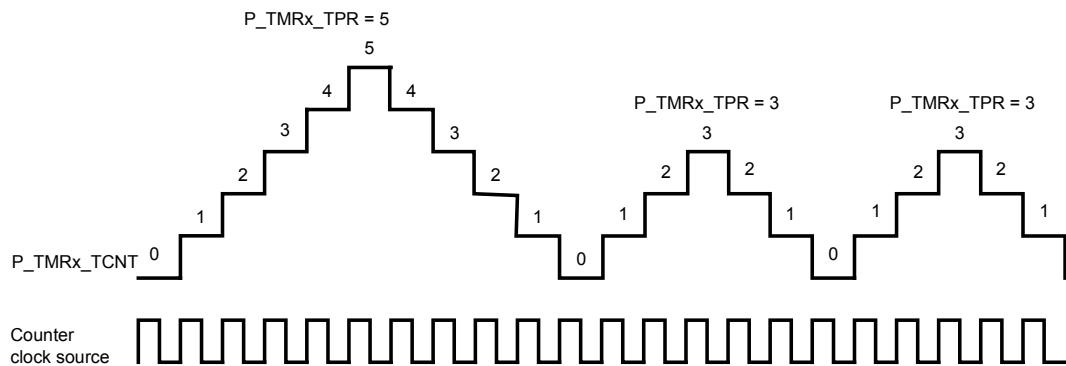


Figure 10-5 Continuous up/down counting example (CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

The initial value of the timer period register can be any value from 0x0000 to 0xFFFF. When the value of the timer counter register equals to timer period register, the PDC timer start to count down to zero. The period, underflow, overflow interrupts behaves the same manner as described in the continuous up counting mode.

The counting direction is recorded at TCDF bit in the P_TMRx_Status (x = 0, 1) register. Either the external clock input pin or internal clock source FCK can be selected as the clock source of the timer. Figure 11-6 shows the center -aligned mode PWM at continuous up/down counting mode of timer 0. And Figure 11-7 indicates the center-aligned PWM timing details.

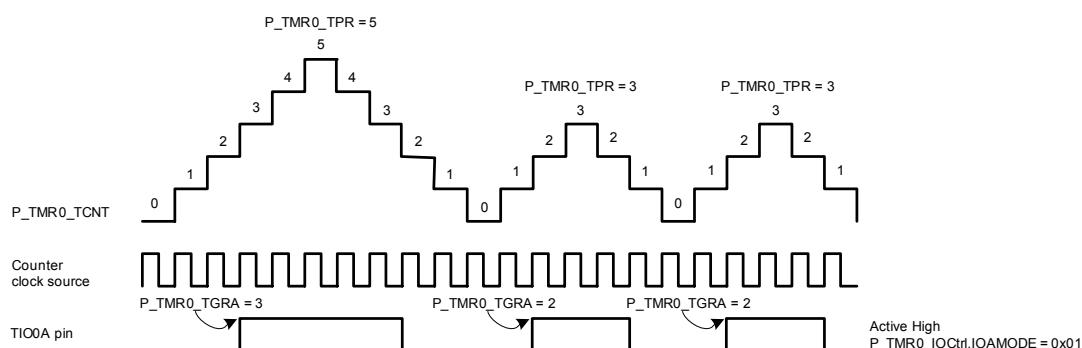


Figure 10-6 Center-Aligned mode PWM

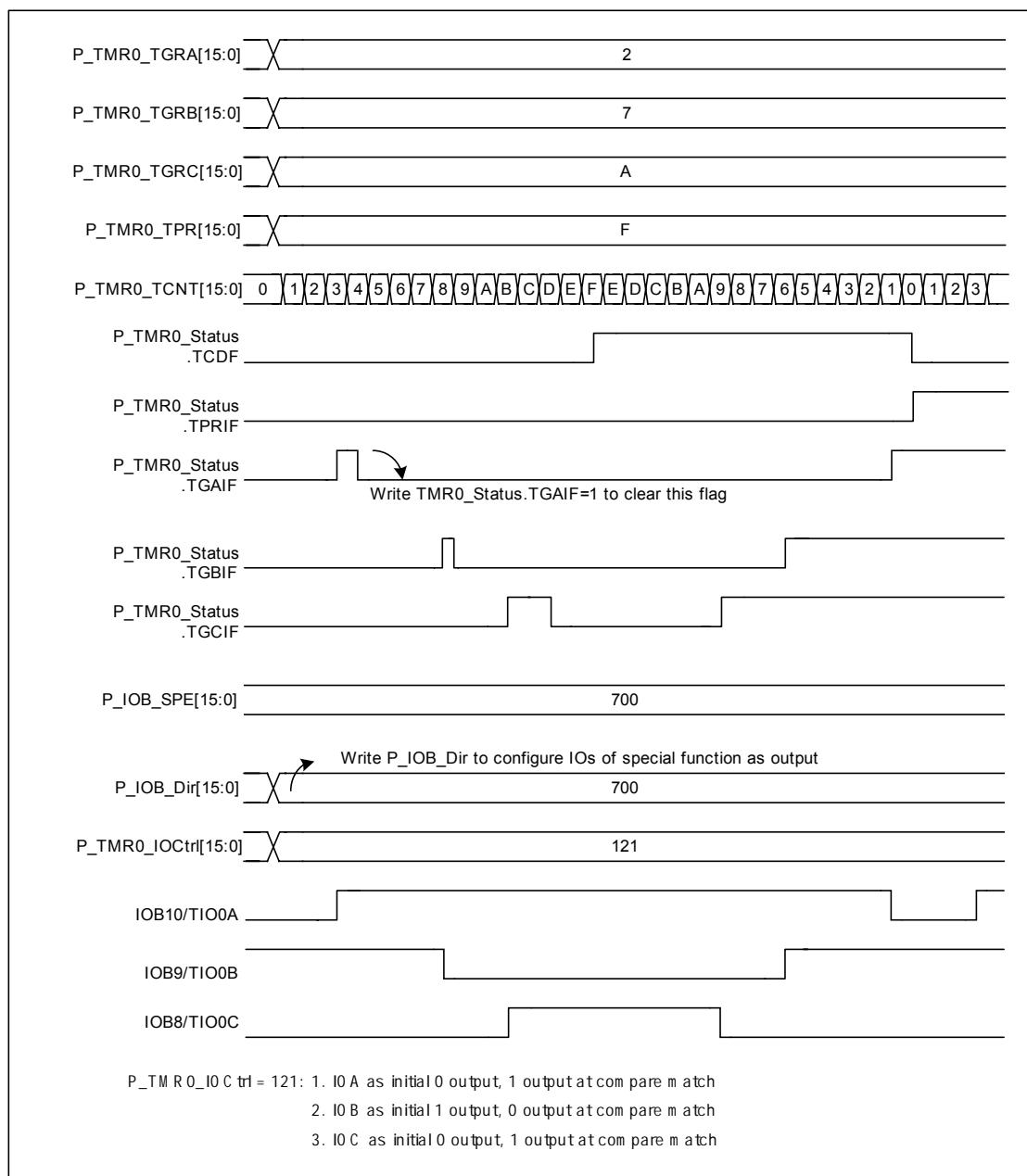


Figure 10-7 Center-Aligned PWM timing of Timer 0

10.5 Registers Descriptions

The PDC timers have the following registers:

10.5.1 Common

Timer counter start register (P_TMR_Start).

10.5.2 PDC Timer 0

1. Timer 0 control register (P_TMR0_Ctrl).
2. Timer 0 input/output control register (P_TMR0_IOCtrl).
3. Timer 0 interrupt enable register (P_TMR0_INT).
4. Timer 0 interrupt status register (P_TMR0_Status).
5. Timer 0 position detection control register (P_POS0_DectCtrl).
6. Timer 0 position detection data register (P_POS0_DectData).
7. Timer 0 counter register (P_TMR0_TCNT).
8. Timer 0 period register (P_TMR0_TPR).
9. Timer 0 general register A (P_TMR0_TGRA).
10. Timer 0 general register B (P_TMR0_TGRB).
11. Timer 0 general register C (P_TMR0_TGRC).
12. Timer 0 buffer register A (P_TMR0_TBRA).
13. Timer 0 buffer register B (P_TMR0_TBRB).
14. Timer 0 buffer register C (P_TMR0_TBRC).

10.5.3 PDC Timer 1

1. Timer 1 control register (P_TMR1_Ctrl).
2. Timer 1 input/output control register (P_TMR1_IOCtrl).
3. Timer 1 interrupt enable register (P_TMR1_INT).
4. Timer 1 interrupt status register (P_TMR1_Status).
5. Timer 1 position detection control register (P_POS1_DectCtrl).
6. Timer 1 position detection data register (P_POS1_DectData).
7. Timer 1 counter register (P_TMR1_TCNT).
8. Timer 1 period register (P_TMR1_TPR).
9. Timer 1 general register A (P_TMR1_TGRA).
10. Timer 1 general register B (P_TMR1_TGRB).
11. Timer 1 general register C (P_TMR1_TGRC).
12. Timer 1 buffer register A (P_TMR1_TBRA).
13. Timer 1 buffer register B (P_TMR1_TBRB).
14. Timer 1 buffer register C (P_TMR1_TBRC).

10.6 Timer 0 and 1 Control Registers

The P_TMRx_Ctrl ($x = 0, 1$) configures the selection of timer clock source, counter clock edge, counter clear source, counter clear edge, capture input sample clock and timer operating modes. TCLKA, TCLKB clock input will be sampled by system clock FCK. Any pulse narrower than four sampling clocks will be ignored. When programmed at counting on both edge, the input clock is halved. Each timer channel can be configured as edge-aligned PWM mode with PWM output or normal operation without any output waveform by setting MODE bits. When MODE bits are set to phase counting mode, the counting phase input is TCLKA/TCLKB on timer 0 and TCLKC/TCLKD on timer 1. The time clock source should be assigned to internal clock in phase counting mode.

10.6.1 P_TMR0_Ctrl (0x7400): Timer 0 Control Register

10.6.2 P_TMR1_Ctrl (0x7401): Timer 1 Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK	MODE						CLEGS

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS	CKEGS						TMRPS

Bit 15:14 **SPCK**: Capture input sample clock select. These bits select the capture input sample clock.

Capture input will be sampled with sample clock. Pulses shorter than four sample clocks will be considered invalid, and will be ignored.

00 = FCK/1

01 = FCK/2

10 = FCK/4

11 = FCK/8

Bit 13:10 **MODE**: Modes select. These bits are used to select the timer operation modes.

0000 = Normal operation (continuous counter up counting)

0100 = Phase counting mode 1

0101 = Phase counting mode 2

0110 = Phase counting mode 3

0111 = Phase counting mode 4

1x0x = Edge-aligned PWM mode (continuous counter up counting, PWM output)

1x1x = Center-aligned PWM mode (continuous counter up/down counting, PWM output)

Bit 9:8 **CLEGS**: Counter clear edge select. These bits select the counter clearing edge

when the clearing source is in input capture mode.

00 = do not clear

01 = rising edge

10 = falling edge

11 = both edge

Bit 7:5 **CCLS**: Counter clear source select. These bits select the TCNT counter clearing source.

000 = TCNT clearing disabled

001 = TCNT cleared by P_TMRx_TGRA (x = 0, 1) capture input

010 = TCNT cleared by P_TMRx_TGRB (x = 0, 1) capture input

011 = TCNT cleared by P_TMRx_TGRC (x = 0, 1) capture input

100 = TCNT cleared by every P_POSx_DectData (x = 0, 1) change 6 times

101 = TCNT cleared by every P_POSx_DectData (x = 0, 1) change 3 times

110 = TCNT cleared by P_POSx_DectData (x = 0, 1) position detection data change

111 = TCNT cleared by P_TMRx_TPR (x = 0, 1) compare match

Bit 4:3 **CKEGS**: Clock edge select, These bits select the input clock edge. When the input clock is counted using both edges, the input clock period is halved. When FCK/1 is selected as counter clock, counter will count at rising edge if count at both edges is selected.

00 = Count at rising edge

01 = Count at falling edge

1X = Count at both edges

Bit 2:0 **TMRPS**: Timer pre-scalar select. These bits select the TCNT counter clock source. It can be selected independently for each channel.

000 = Counts on FCK /1

001 = Counts on FCK /4

010 = Counts on FCK /16

011 = Counts on FCK /64

100 = Counts on FCK /256

101 = Counts on FCK /1024

110 = Counts on TCLKA pin input, maximum external clock source input is 3.0 MHz.

111 = Counts on TCLKB pin input, maximum external clock source input is 3.0 MHz.

10.7 Timer 0 and 1 Input and Output Control Register

The P_TMRx_IOCrl (x =0, 1) register controls the PWM output and input capture action type of TIOxA, TIOxB, and TIOxC (x = 0, 1) pins. By setting the CCLS and MODE bits in P_TMRx_Ctrl (x = 0, 1) register will determine the timer IO action mode. When choosing PWM output mode, the IOAMODE/IOBMODE/OCMODE bits determines the waveform generation depending on the active clock edge. When choosing input capture mode, the IOAMODE/IOBMODE/OCMODE bits defines the capture event including position detection changed.

10.7.1 P_TMR0_IOC (0x7410) : Timer 0 IO control register

10.7.2 P_TMR1_IOC (0x7411) : Timer 1 IO control register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved				IOCMODE			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

Bit 15:12: Reserved

Bit 11:8: **IOCMODE**: Select Timer 0/Timer 1 IOC Configuration

Bit 7:4: **IOBMODE**: Select Timer 0/Timer 1 IOB Configuration

Bit 3:0: **IOAMODE**: Select Timer 0/Timer 1 IOA Configuration

PWM compare match output mode :

0000 = Initial output 0, 0 output at compare match

0001 = Initial output 0, 1 output at compare match

0010 = Initial output 1, 0 output at compare match

0011 = Initial output 1, 1 output at compare match

01xx = Output hold

Input capture mode :

1000 = Issue input capture interrupt at rising edge

1001 = Issue input capture interrupt at falling edge

101x = Issue input capture interrupt at both edges

11xx = Input capture when Position Detection Register changes

(capture TCNT register value to TGR register) and issue interrupt (for TGRA register only).

10.8 Timer 0 and 1 Interrupt Enable Register

The P_TMRx_INT (x = 0, 1) register is used to enable or disable A/D conversion start request by TGRA compare match, interrupt requests for position detection changes, overflow/underflow of TCNT, period register compare match and input capture/compare match of TGRA, TGRB, TGRC.

10.8.1 P_TMR0_INT (0x7420): Timer 0 Interrupt Enable Register

10.8.2 P_TMR1_INT (0x7421): Timer 1 Interrupt Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
Reserved							PDCIE

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	TCUIE	TCVIE	TPRIE	Reserved	TGCIE	TGBIE	TGAIE

Bit 15:9 Reserved

Bit 8 **PDCIE**: Position detection change interrupt enable bit. Enables or disables interrupt request by position detection data register P_POSx_DectData (x = 0, 1) change.

0 = Disable

1 = Enable

Bit 7 **TADSE**: A/D conversion start request enable bit. Enables or disables generation of A/D conversion start request by TGRAreregister compare match.

0 = Disable

1 = Enable

Bit 6 **TCUIE**: Underflow interrupt enable bit. Enables or disables interrupt request by counter underflow.

0 = Disable

1 = Enable

Bit 5 **TCVIE**: Overflow interrupt enable bit. Enables or disables interrupt request by counter overflow.

0 = Disable

1 = Enable

Bit 4 **TPRIE**: Timer Period Register interrupt enable bit. Enables or disables interrupt request by TPR register compare match.

0 = Disable

1 = Enable

Bit 3 Reserved

Bit 2 **TGCIE**: Timer General C Register interrupt enable bit. Enables or disables interrupt request by TGRC register input capture or compare match.

0 = Disable

1 = Enable

Bit 1 **TGBIE**: Timer General B Register interrupt enable bit. Enables or disables interrupt request by TGRB register input capture or compare match.

0 = Disable

1 = Enable

Bit 0 **TGAIE:** Timer General A Register interrupt enable bit. Enables or disables interrupt request by TGRA register input capture or compare match.

0 = Disable

1 = Enable

10.9 Timer 0 and 1 Interrupt Status Register

The interrupt status register indicates the event generation of position detection changes, an underflow/overflow of TCNT, period register compare match and input capture/compare match of TGRA, TGRB, TGRC. These flags show the interrupt sources. An interrupt would be generated when the corresponding interrupt enable bit is set in P_TMRx_INT (x = 0, 1) register.

The TCDF represents the counter direction when timer is setup to center-aligned PWM mode or phase counting mode.

10.9.1 P_TMR0_Status (0x7425): Timer 0 Interrupt Status Register

10.9.2 P_TMR1_Status (0x7426): Timer 1 Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
Reserved							PDCIF

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	TCUIF	TCVIF	TPRIF	Reserved	TGCIF	TGBIF	TGAIF

Bit 15:9 Reserved

Bit 8 **PDCIF:** Position detection changes interrupt flag. Status flag that indicates position detection register change. The flag will be cleared if '1' has been written to.

0 = Position no changed

1 = Position changed

Bit 7 **TCDF:** Timer Counter Count direction flag. Status flag that shows the counting direction in which TCNT counts.

0 = Up-counting

1 = Down-counting

Bit 6	TCUIF: Timer Counter Underflow flag. Status flag shows that TCNT underflow has occurred when Timer 0 and 1 are set to phase counting mode. The flag will be cleared if '1' has been written to. 0 = Underflow not occurred 1 = Underflow has occurred
Bit 5	TCVIF: Timer Counter Overflow flag. Status flag shows that TCNT overflow has occurred. The flag will be cleared if '1' has been written to. 0 = Overflow not occurred 1 = Overflow has occurred
Bit 4	TPRIF: Timer Period Register compare match flag. This status flag indicates a TPR register compare match event has been occurred. Writing '1' will clear this flag. 0 = Compare match not occurred 1 = Compare match has occurred
Bit 3	Reserved
Bit 2	TGCIF: Timer General C Register input capture/compare match flag. This status flag indicates a TGRC register input capture or compare match event has been occurred. Writing '1' will clear this flag. 0 = Input capture/compare match not occurred 1 = Input capture/compare match has occurred
Bit 1	TGBIF: Timer General B Register input capture/compare match flag. This status flag indicates a TGRB register input capture or compare match event has been occurred. Writing '1' will clear this flag. 0 = Input capture/compare match not occurred 1 = Input capture/compare match has occurred
Bit 0	TGAIF: Timer General A Register input capture/compare match flag. This status flag indicates a TGRA register input capture or compare match event has been occurred. Writing '1' will clear this flag. 0 = Input capture/compare match not occurred 1 = Input capture/compare match has occurred

10.10 Timer Start Register

The P_TMR_Start register selects the operation of counter start/stop for the P_TMRx_TCNT ($x = 0 \sim 4$). When counter operation stopped, its contents will be cleared. Set TMR0ST or TMR1ST bit to 1 would start the P_TMR0_TCNT or P_TMR1_TCNT register immediately and vice versa.

10.10.1 P_TMR_Start (0x7405): Timer Counter Start Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST

Bit 15:5 Reserved

Bit 4 **TMR4ST:** Timer 4 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

Bit 3 **TMR3ST:** Timer 3 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

Bit 2 **TMR2ST:** Timer 2 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

Bit 1 **TMR1ST:** Timer 1 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

Bit 0 **TMR0ST:** Timer 0 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

10.11 Timer 0 and 1 Position Detection Control Register

There are two position detection control registers available in SPMC75X family MCU : P_POS0_DectCtrl and P_POS1_DectCtrl for timer 0 and timer 1 respectively. The control-registers control the sampling settings of position detection signals from TIOxA, TIOxB and TIOxC (x = 0, 1) input pins. The sampling parameters such as sampling clock, valid sampling count select, and sampling delay are all programmable.

The SPLMOD bits determine the sampling position signal condition. They can be selected from three modes : sample when PWM on, sample regularly, or sample while low side transistors are in conducting current. The SPLCNT bits select the sampling delay and used in modes where sampling is made while PWM is on or lower side phase are conducting current. It helps to prevent erroneous detection due to the glitch that occurs immediately after the transistor is on.

10.11.1 P_POS0_DectCtrl (0x7462): Timer 0 Position Detection Control Register
10.11.2 P_POS1_DectCtrl (0x7463): Timer 1 Position Detection Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPLCK		SPLMOD			SPLCNT		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PDEN	SPDLY						

Bit 15:14 **SPLCK**: Sampling clock select. Select FCK/4, FCK/8, FCK/32, or FCK/128 for position sampling clock

00 = FCK/4

01 = FCK/8

10 = FCK/32

11 = FCK/128

Bit 13:12 **SPLMOD**: Sampling mode select. Select one of three modes: sampling when PWM signal is active (PWM is on), sampling regularly, or sampling when lower side (UN, VN, WN) phases are conducting current.

00 = Start sampling when one of Ux/Vx/Wx/UxN/VxN/WxN (x=1,2) output is active and delay counter count to the value of SPDLY

01 = Sample immediately regardless of delay counter

10 = Start sampling when one of UxN/VxN/WxN (x=1,2) output is active and delay counter count to the value of SPDLY

11 = Reserved

Bit 11:8 **SPLCNT**: Sampling count select. These bits select the sampling count for the valid external position detection signals. The position signals must be sampled continuously match as many times as the sampling count set, for the position signals to be considered valid. The valid settings are from 1 to 15 times. Note that count 0 and 1 are assumed to be one time.

Bit 7 **PDEN**: Position detection enable. This bit enables/disables the position detection function for position input pins TIOA~C. When enabled, the input signals of these pins will be sampled and the results will be latched to PDR [2:0] bits in POS_DectData register. When disabled, PDR [2:0] will remain its status.

0 = Disable

1 = Enable

Bit 6:0 **SPDLY:** Sampling delay. These bits set a delay time clock in which at SPLCK clock source. It is used to delay sampling in order to prevent erroneous detection due to noise that occurs immediately after any PWM output is active. The delay counter start counting when one of Ux/Vx/Wx/UxN/VxN/WxN (x=1,2) output is active.

10.12 Timer 0 and 1 Position Detection Data Register

The current filtered position data will be latched to these registers. The sampling settings can be set in position detection control registers P_POSx_DectCtrl (x = 0, 1).

10.12.1 P_POS0_DectData (0x7464): Timer 0 Position Detection Data Register

10.12.2 P_POS1_DectData (0x7465): Timer 1 Position Detection Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved						PDR	

Bit 15:13 Reserved

Bit 3:0 **PDR**

PDR[2] = Noise filtered position detection input from pin TIO0C

PDR[1] = Noise filtered position detection input from pin TIO0B

PDR[0] = Noise filtered position detection input from pin TIO0A

10.13 Timer 0 and 1 Counter Register

The PDC timer has two TCNT counters (P_TMR0_TCNT and P_TMR1_TCNT), one for each channel.

The TCNT counters are 16-bit readable registers that increment/decrement according to input clocks.

Bits TMRPS in corresponding timer control register can select input clocks. P_TMR0_TCNT and P_TMR1_TCNT increment/decrement in center-aligned PWM mode, while they only increment in other modes. The TCNT counters are initialized to 0x0000 by compare matches with corresponding TGRA, TGRB, TGRC, or input captures to TGRA, TGRB, TGRC, or P_POSx_DectData (x = 0 , 1) data changes. When the TCNT counters overflow, an TCUIF flag in timer interrupt status register for the corresponding channel is set to 1. When TCNT underflows, a TUDIF flag in timer interrupt status register is set to 1.

10.13.1 P_TMR0_TCNT (0x7430): Timer 0 Counter Register

10.13.2 P_TMR1_TCNT (0x7431): Timer 1 Counter Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

10.14 Timer 0 and 1 General and Buffer Register

TGRA, TGRB, TGRC are 16-bit registers. PDC Timer has six timer general registers, three for each channel. The TGR registers are dual function 16-bit readable/writable registers, functioning as either PWM output or input capture registers.

The values in TGR and TCNT are constantly compared with each other when the TGR registers are used as PWM output registers. When the both values match, the TGAIF, TGBIF, TGCIF bits in corresponding timer interrupt status register are set to 1. Compare match outputs can be selected by TIOxA, TIOxB and TIOxC ($x = 0, 1$). When the TGR registers are used as input capture registers, the TCNT value is stored after detecting external signals. At this point, TGAIF, TGBIF, TGCIF bits in the corresponding timer interrupt status register are set to 1. Detection edges for input capture signals can be selected by TIOxA, TIOxB and TIOxC ($x = 0, 1$) active status and programmable via CCLS bits in P_TMRx_Ctrl ($x = 0, 1$) register.

When PWM mode, edge-aligned PWM mode, or center-aligned PWM mode is selected, the TGR register behaves as the duty ratio value register. Upon reset, the TGR registers are initialized to 0x0000.

When bits CCLS are set to 100'b, 101'b, 110'b, the PDC timer behaves as PDC mode used for BLDC motor application. The hall position signals are connected to TIOxA, TIOxB, TIOxC ($x = 0, 1$). The TCNT register is stored to TGRA register according to bits value of CCLS and CLEG bits should be assigned to clear on both edge. When position detection change event occurred, the TCNT register will be latched to TGRA then reset to 0x0000. User could use this information to read the correct TGRA value to calculate the motor speed.

The timer buffer registers TBRA, TBRB and TBRC are the double buffers of TGRA, TGRB and TGRC, respectively. The value of TGRx (x=A, B, C) can automatically be updated when the period compare match event occurs. That is, the duty ratio value will not be updated until one period ends completely. When the TBR registers are used as input capture registers, the TCNT value is stored at the falling edge of input capture port.

10.14.1 P_TMR0_TGRA (0x7440): Timer 0 General Register A

10.14.2 P_TMR0_TGRB (0x7441): Timer 0 General Register B

10.14.3 P_TMR0_TGRC (0x7442): Timer 0 General Register C

10.14.4 P_TMR1_TGRA (0x7443): Timer 1 General Register A

10.14.5 P_TMR1_TGRB (0x7444): Timer 1 General Register B

10.14.6 P_TMR1_TGRC (0x7445): Timer 1 General Register C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

10.14.7 P_TMR0_TBRA (0x7450): Timer 0 Buffer Register A

10.14.8 P_TMR0_TBRB (0x7451): Timer 0 Buffer Register B

10.14.9 P_TMR0_TBRC (0x7452): Timer 0 Buffer Register C

10.14.10 P_TMR1_TBRA (0x7453): Timer 1 Buffer Register A

10.14.11 P_TMR1_TBRB (0x7454): Timer 1 Buffer Register B

10.14.12 P_TMR1_TBRC (0x7455): Timer 1 Buffer Register C

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRBUF							

10.15 Timer 0 and 1 Period Register

The P_TMRx_TPR ($x = 0, 1$) is a 16-bit readable/writable register. It is used to set the period of PWM waveform. When P_TMRx_TCNT ($x = 0, 1$) register reaches P_TMRx_TPR ($x = 0, 1$) register value, P_TMRx_TCNT ($x = 0, 1$) register will be cleared to 0x0000 (up-counting mode) or start down-count (continuous up-/down-counting mode) according to MODE bits programmed in P_TMRx_Ctrl ($x = 0, 1$) registers. Its default value is 0xFFFF. When P_TMRx_TPR ($x = 0, 1$) register is set to 0x0000, the P_TMRx_TCNT ($x = 0, 1$) register counter will stop counting and remain at 0x0000.

10.15.1 P_TMR0_TPR (0x7435): Timer 0 Period Register

10.15.2 P_TMR1_TPR (0x7436): Timer 1 Period Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

10.16 PDC Timers Operation

10.16.1 Normal Counting Operation

When TMR0ST or TMR1ST bit is set in P_TMR_Start register, the P_TMRx_TCNT ($x = 0, 1$) register for the corresponding channel beginning up direction counting. The counter register behaves as a free running operation and would be reset to 0x0000 when reaches 0xFFFF. Through configures the CCLS bits to 111'b, the timers is setup as periodic counter. The counter register would be reset to 0x0000 when reaches the value of P_TMRx_TPR ($x = 0, 1$) registers. Figure 11-8 shows the programming flowchart of normal counting operation.

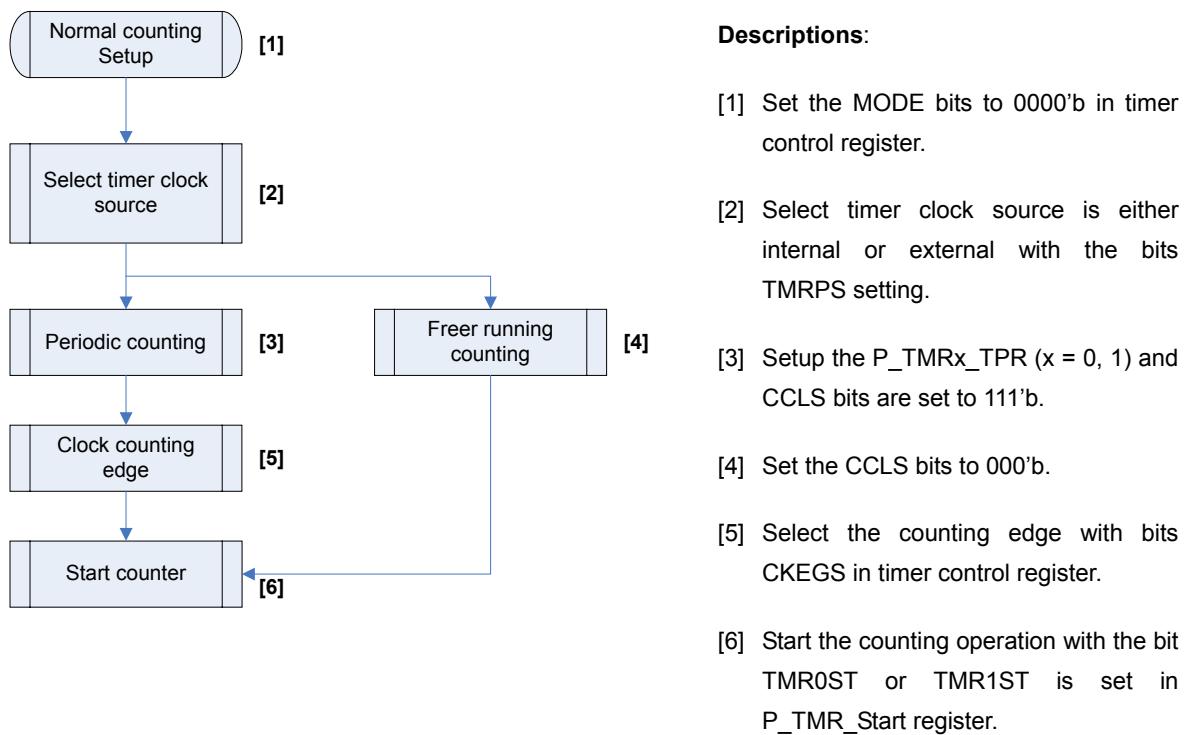


Figure 10-8 Example programming flowchart of normal counting operation

The initial value of P_TMRx_TCNT ($x = 0, 1$) is 0x0000 and P_TMRx_TPR ($x = 0, 1$) is 0xFFFF. When the corresponding bit TMR0ST or TMR1ST is set to 1, the P_TMRx_TCNT ($x = 0, 1$) behaves as the free-running counter. When the counter overflows, the TCVIF flag in P_TMRx_Status ($x = 0, 1$) is set to 1. If the bit value of TCVIE in P_TMRx_INT ($x = 0, 1$) register is set to 1, the PDC timer requests an interrupt. After overflow, the P_TMRx_TCNT ($x = 0, 1$) register starts an increment operation again from 0x0000.

When a period compare match event is selected as P_TMRx_TCNT ($x = 0, 1$) register clearing source, the counter behaves as the periodic counting operation. The P_TMRx_TPR ($x = 0, 1$) registers for setting the period the period and counter-clearing source is selected by setting CCLS bits to 111'b. After the settings have made, the counter register start an increment operation as periodic counter when corresponding bit TMR0ST or TMR1ST is set to 1. When counter matches the value of P_TMRx_TPR ($x = 0, 1$) register, the TPRIF flag is set to 1 in timer interrupt status register and counter register is reset to 0x0000. If the bit value of TPRIE in P_TMRx_INT ($x = 0, 1$) register is set to 1, the PDC timer requests an interrupt.

10.16.2 PWM Compare Match Output Operation

The PDC timer module has two channels and can perform PWM compare match output function up to six pins output. The output waveforms have active low at compare match, active high at compare match and output hold for the corresponding TIOxA, TIOxB, TIOxC (x = 0, 1) output pin using compare match with P_TMRx_TGRA, P_TMRx_TGRB, P_TMRx_TGRC (x = 0, 1) register respectively. Figure 11-9 shows the programming flowchart of PWM compare match output operation. Figure 11-10 shows the compare match output and interrupt timing waveform.

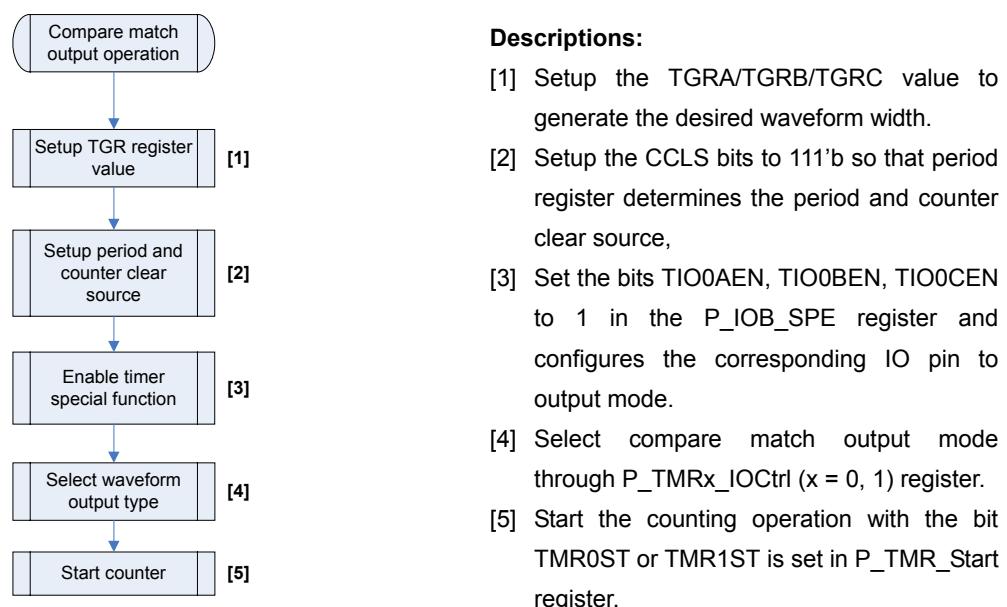


Figure 10-9 Example programming flowchart of PWM compare match output operation

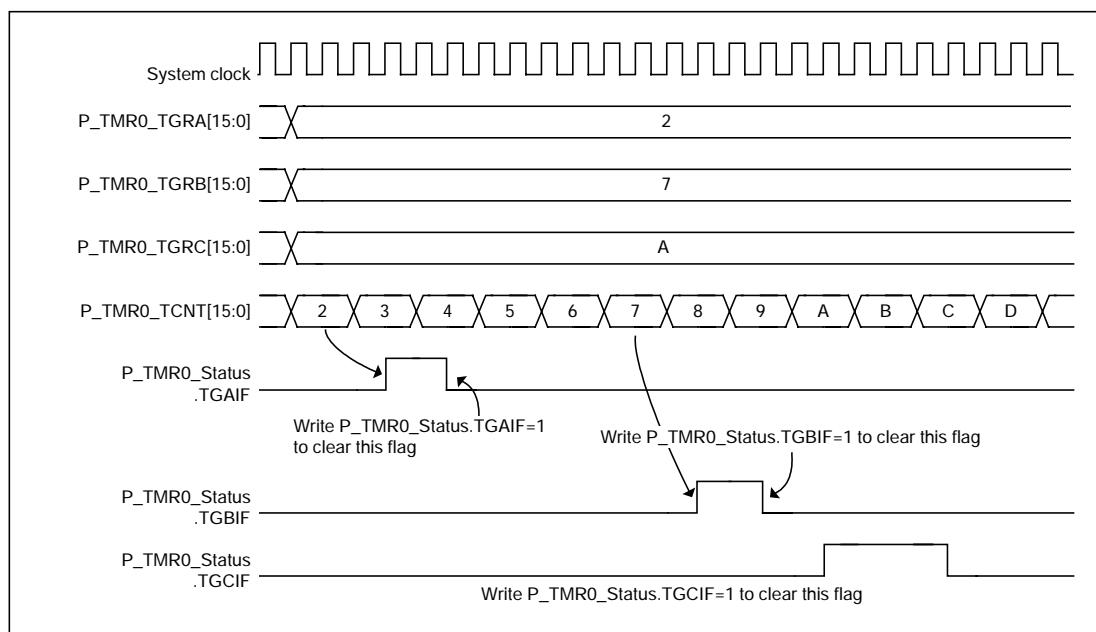


Figure 10-10 Compare match and interrupt timing

10.16.3 Input Capture Operation

The capture function activation and the input edge at which the interrupt status flag issued are determined by the bits of IOAMOD, IOBMOD and IOCMod in the P_TMRx_IOCctrl (x = 0, 1) register, respectively. It can be the rising edge, falling edge or both edge. The value of counter is always transferred to TGRx (x=A, B, C) and TBRx (x=A, B, C) at the rising and falling edge of corresponding input capture port, respectively. The counter register, P_TMRx_TCNT (x = 0, 1) can be cleared according to the setting of CCLS in P_TMRx_Ctrl (x = 0, 1) register. The counter clear source can be one of TIO0A, TIO0B and TIO0C at the selected edge according to CLEGS in P_TMRx_Ctrl (x = 0, 1). The powerful and flexible input capture function provides the essential feature for the motor control.

Table 11-2 shows the input capture configurations settings and results. When the input capture function is selected, the pulse width or period on input pin can be measured. Figure 11-13 shows the programming flowchart of input capture operation. Figure 11-12 is an example of input capture TIO0x (x=A, B, C). The correlations between the configuration of P_TMR0_IOCctrl and interrupt even are shown.

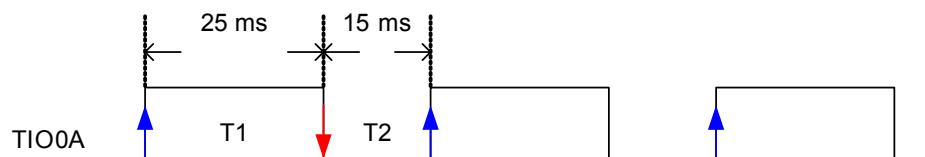


Figure 10-11 input capture signal connected to TIO0A

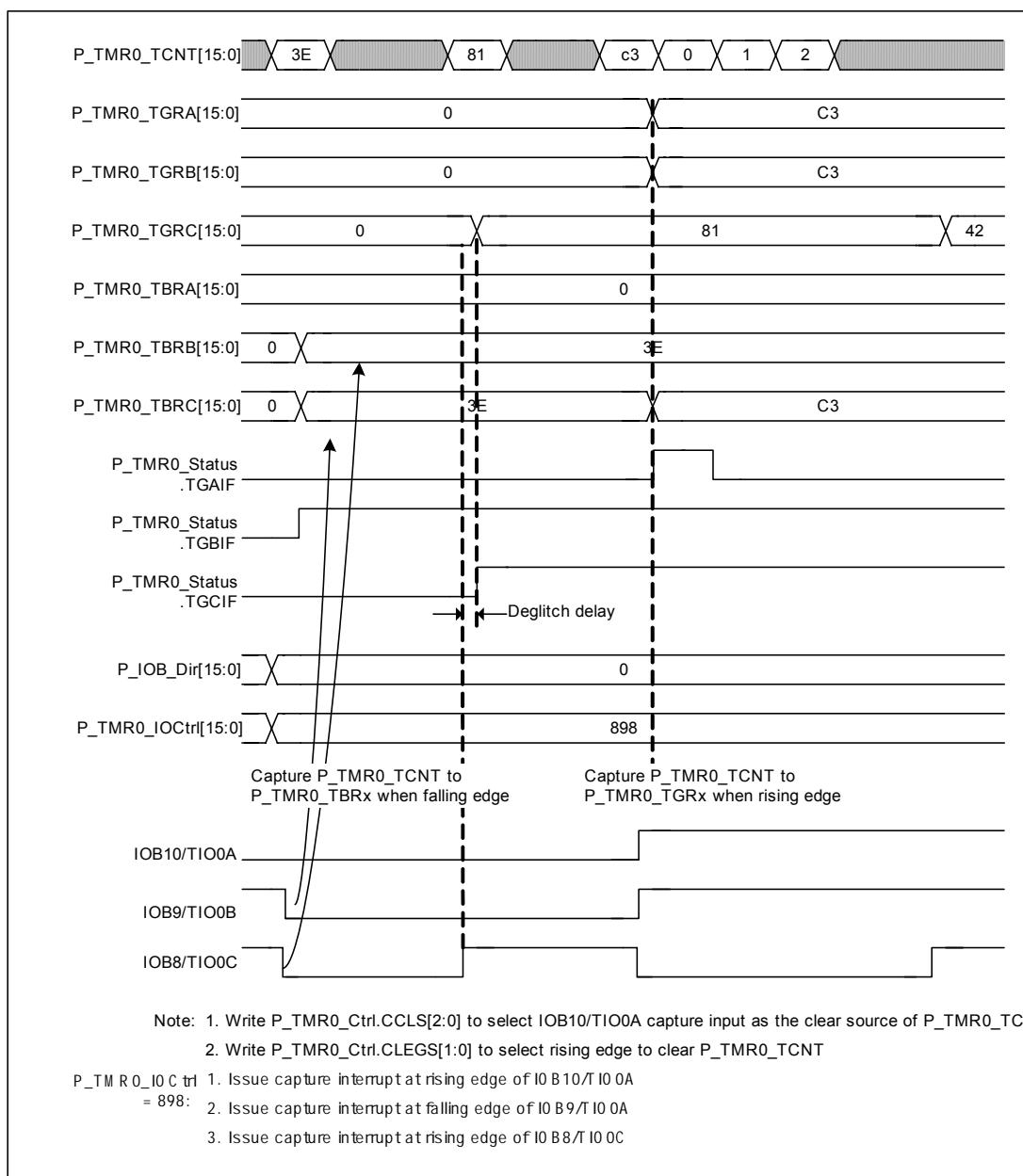


Figure 10-12 Capture input signal pulse width and pulse cycle

Table 11-2 input capture configuration settings and results

Input capture settings			Enter interrupt at signal edge	Results
CLEGS	CCLS	IOAMODE		
Rising edge	TGRA	Rising edge	Rising edge	P_TMR0_TGRA = period (40 ms) P_TMR0_TBRA = T1 (25ms)
Rising edge	TGRA	Falling edge	Falling edge	P_TMR0_TGRA = period (40 ms) P_TMR0_TBRA = T1 (25ms)
Rising edge	TGRA	Both edge	Both edge	P_TMR0_TGRA = period (40 ms) P_TMR0_TBRA = T1 (25ms)
Falling	TGRA	Rising	Rising edge	P_TMR0_TGRA = T2 (15 ms)

Input capture settings			Enter interrupt at signal edge	Results
CLEGS	CCLS	IOAMODE		
edge		edge		P_TMR0_TBRA = period (40ms)
Falling edge	TGRA	Falling edge	Falling edge	P_TMR0_TGRA = T2 (15 ms) P_TMR0_TBRA = period (40ms)
Falling edge	TGRA	Both edge	Both edge	P_TMR0_TGRA = T2 (15 ms) P_TMR0_TBRA = period (40ms)
Both edge	TGRA	Rising edge	Rising edge	P_TMR0_TGRA = T2 (15 ms) P_TMR0_TBRA = T1 (25ms)
Both edge	TGRA	Falling edge	Falling edge	P_TMR0_TGRA = T2 (15 ms) P_TMR0_TBRA = T1 (25ms)
Both edge	TGRA	Both edge	Both edge	P_TMR0_TGRA = T2 (15 ms) P_TMR0_TBRA = T1 (25ms)

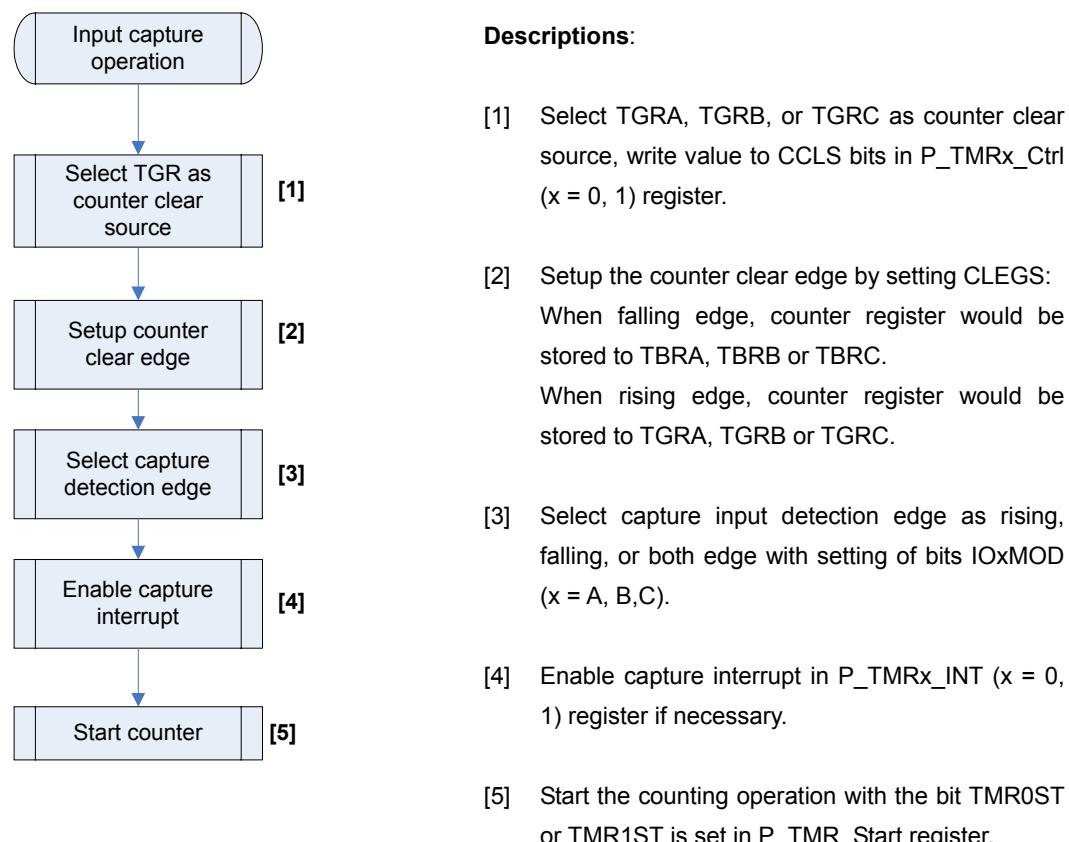


Figure 10-13 Example programming flowchart of input capture operation

10.16.4 Position Detection Change (PDC) Mode Operation

The PDC timer has an extremely useful feature for position detection control used in BLDC motor driving application. The P_TMRx_TCNT ($x = 0, 1$) value can be transferred to TGRA, when CCLS bits is set to 100'b, 101'b or 110'b in the P_TMRx_Ctrl ($x = 0, 1$) register. Through the value of CCLS bits, the counter register can be store to TGRA every six, three, one position detection data changes. Whenever the position detection changed event occurs, the counter register will be reset to 0x0000 after transferred to TGRA and PDCIF interrupt flag is set to 1. If the position detection interrupt enable bit PDCIE is set to 1 in the corresponding P_TMRx_INT ($x = 0, 1$) register, it would request a PDC interrupt to CPU.

Through programming the bits value of SPLCNT, SPLCK and SPLMOD, user could avoid the noise on the hall signal inputs and position detection data register P_POS0_DectData, P_POS1_DectData can latch the correct position data. Figure 11-14 shows the programming flowchart of PDC mode operation. Figure 11-15 indicates the timing for position detection mechanism with noise filter.

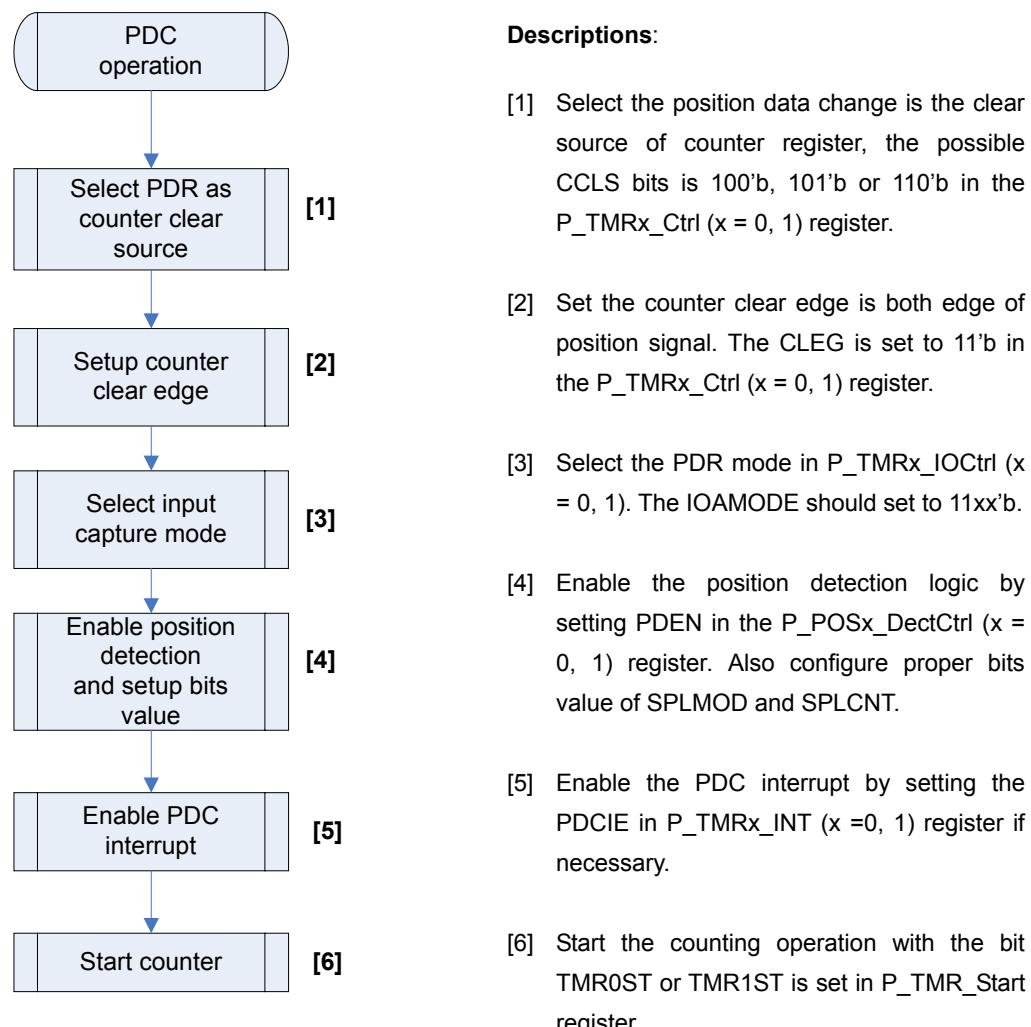


Figure 10-14 Example programming flowchart of PDC operation

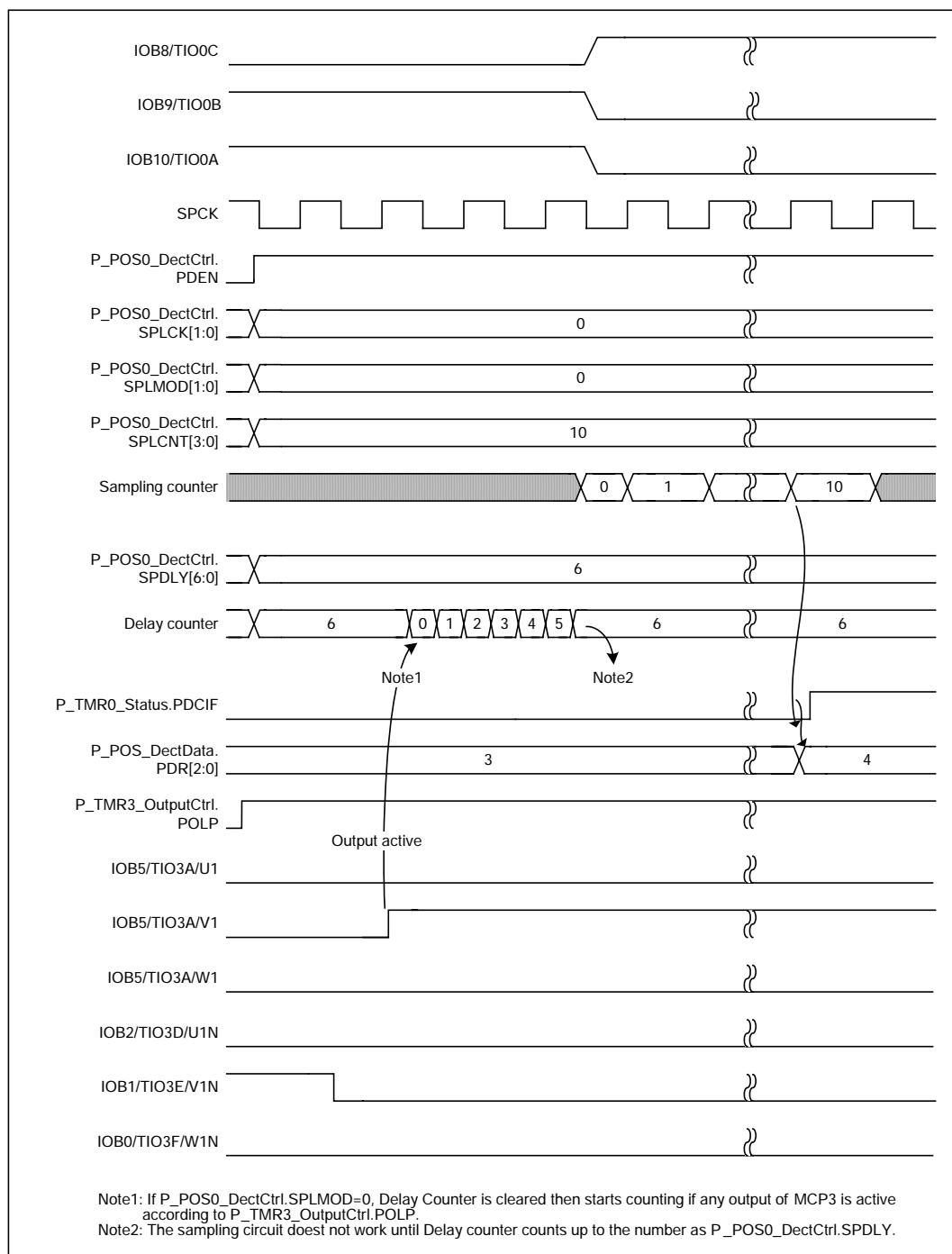


Figure 10-15 Position detection with noise filter

10.16.5 Phase Counting Mode Operation

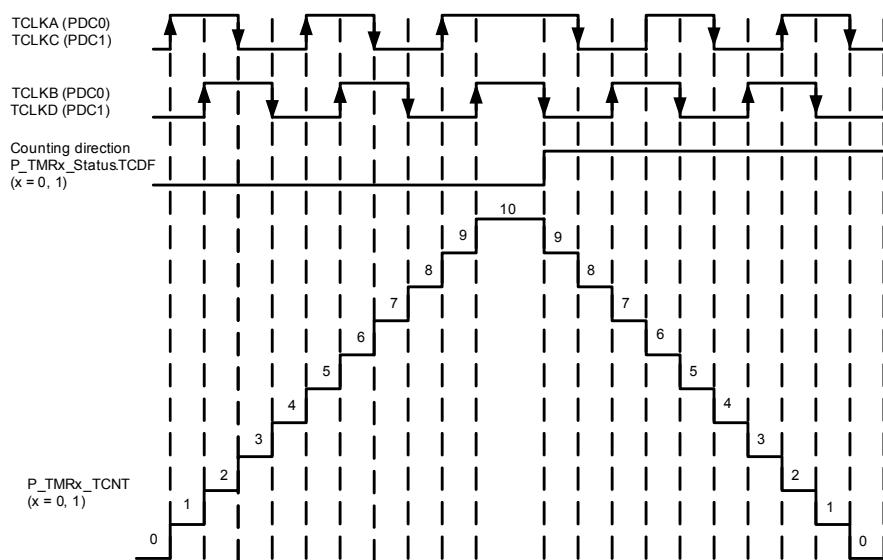
In phase counting mode, the phase difference between two external clock inputs is detected and timer counter counts up or down according to the clock phase relationship. This mode can be set for PDC0 and PDC1. The general application is for two-phase quadrature encoder pulse inputs. The clock source of channel 0 utilizes TCLKA and TCLKB pins. Channel 1 utilizes TCLKC and TCLKD pins. The SPMC75X family MCU supports the following four modes directional phase counting operation. Figure 11-16 to 11-19 represents the four-phase counting mode operation, and Figure 11-20 shows the programming flowchart of phase counting mode procedure.

10.16.6 Phase Counting Mode 1

In phase counting mode 1, the P_TMRx_TCNT ($x = 0, 1$) always counts up as long as the TCLKB/TCLKD clock source is leading 90 degree with TCLKA/TCLKC. On the other hand, the P_TMRx_TCNT always count down when TCLKB/TCLKD clock source is lagging 90 degree with TCLKA/TCLKC. This mode is useful for encoder equipped motor drive application. Table 11-3 shows phase counting mode 1 relationship. The phase resolution is amplified four times compared to encoder resolution specification (pulse / revolution). Figure 11-16 shows the example of phase counting 1.

Table 11-3 phase counting mode 1 relationship

TCLKA (PDC0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	Counting Operation
H	Rising	Up-count
L	Falling	
Rising	L	
Falling	H	
H	Falling	Down-count
L	Rising	
Rising	H	
Falling	L	

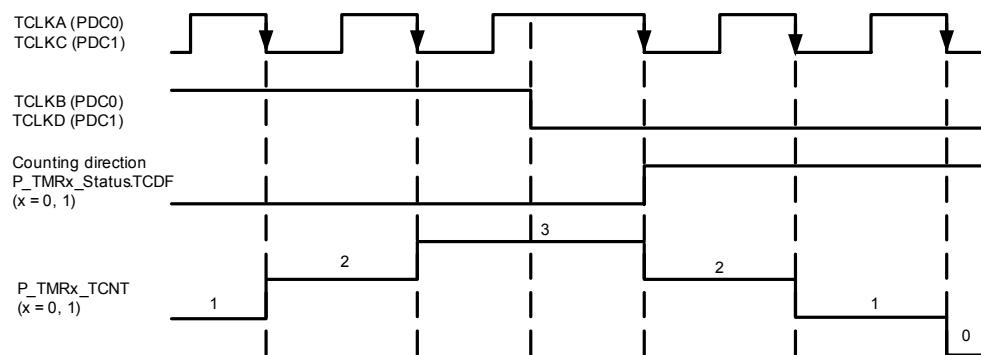

Figure 10-16 phase counting mode 1

10.16.7 Phase Counting Mode 2

In phase counting mode 2, the **P_TMRx_TCNT (x = 0, 1)** counting direction is determined by the logic level of **TCLKB/TCLKD**. When **TCLKB/TCLKD** remains logic 'H' level, the counter does the up-counting operation. If **TCLKB/TCLKD** is logic level 'L', it does the down-counting operation. Table 11-4 shows the relationship. The counting operation is synchronous to the falling edge of **TCLKA/TCLKC**. Figure 11-17 shows the phase counting mode 2 examples.

Table 11-4 phase counting mode 2 relationship

TCLKA (PDC0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	Counting Operation
H	Rising	—
L	Falling	—
Rising	L	—
Falling	H	Up-count
H	Falling	—
L	Rising	—
Rising	H	—
Falling	L	Down-count

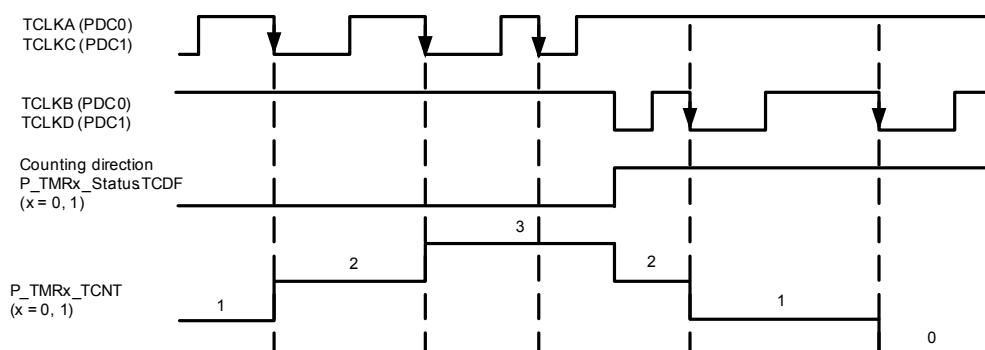

Figure 10-17 phase counting mode 2

10.16.8 Phase Counting Mode 3

In phase counting mode 3, the P_TMRx_TCNT ($x = 0, 1$) does the up counting operation when TCLKB/TCLKD remains at logic level 'H', and synchronous to the falling edge of TCLKA/TCLKC. On the other hand, the P_TMRx_TCNT ($x = 0, 1$) does the down counting operation in the situation of TCLKA/TCLKC at logic level 'H', and synchronous to the falling edge of TCLKB/TCLKD. The following table shows the relationship and Figure 11-18 represents this phase counting example.

Table 11-5 phase counting mode 3 relationships

TCLKA (PDC0) TCLKC (PDC1)	TCLKB (PDC0) TCLKD (PDC1)	Counting Operation
H	Rising	—
L	Falling	—
Rising	L	—
Falling	H	Up-count
H	Falling	Down-count
L	Rising	—
Rising	H	—
Falling	L	—


Figure 10-18 phase counting mode 3

10.16.9 Phase Counting Mode 4

In phase counting mode 4, the P_TMRx_TCNT counting direction is determined by the combinations of logic level and active level selection of TCLKx ($x = A, B, C, D$). When TCLKx ($x = A, C$) is at logic 'H'/'L' level and TCLKy ($y = B, D$) clock is in the rising/falling edge, the counter will do the up counting operation. In the case of TCLKx ($x = A, C$) is at logic 'H'/'L' level and TCLKy ($y = B, D$) clock is in the falling/rising edge; the counter will do the down counting operation. The following table shows the relationship and Figure 11-19 represents this phase counting example.

Table 11-6 phase counting mode 4 relationship

TCLKA (PDC0)	TCLKB (PDC0)	Counting Operation
TCLKC (PDC1)	TCLKD (PDC1)	
H	Rising	Up-count
L	Falling	Up-count
Rising	L	—
Falling	H	—
H	Falling	Down-count
L	Rising	Down-count
Rising	H	—
Falling	L	—

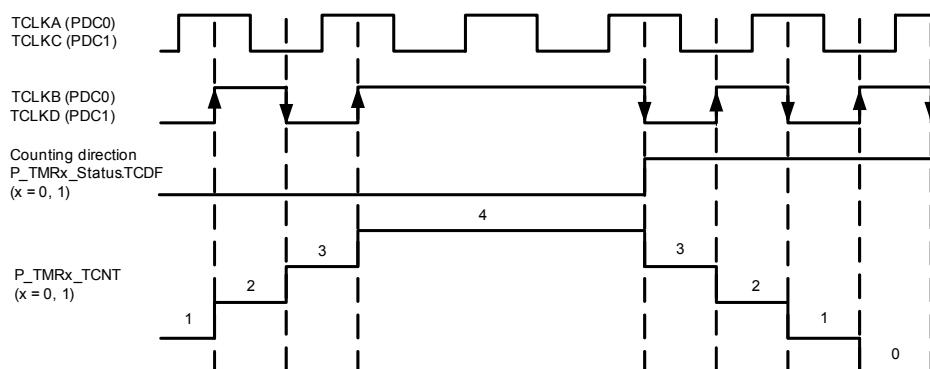


Figure 10-19 phase counting mode 4

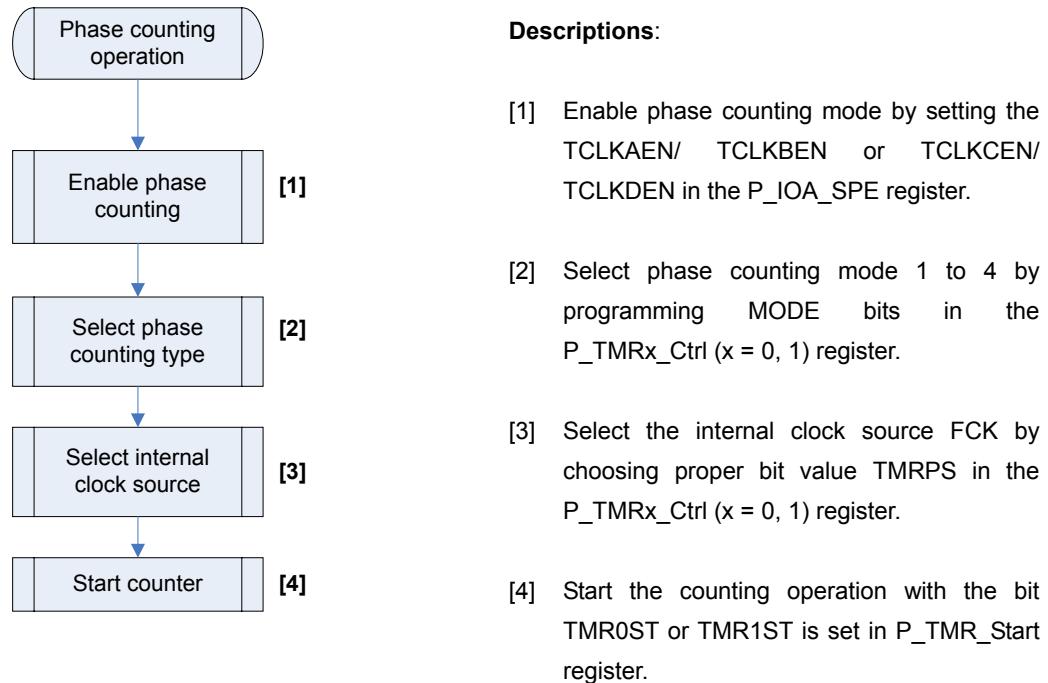


Figure 10-20 Example programming flowchart of phase counting operation

10.17 Application Example Design Tips

【Example 11-1】: PDC position detection and speed calculation flow for BLDC motor driving. The Listing below shows the example programming procedure.

```

/******************************************************************************/
/* Init_PDC0() : initialize PDC0 module, and enable PDC interrupt          */
/*               detection control should set to prevent the glitch signal   */
/******************************************************************************/
void Init_PDC0(void)
{
    P_TMR0_Ctrl->W = CW_TMR0_TMRPS_FCKdiv64 | CW_TMR0_CCLS_PDR6 | \
                      CW_TMR0_CLEGS_Both | CW_TMR0_MODE_Normal;
    /* count on rising, clear on every six PDR change */
    /* count on FCK/64, rising edge */
    /* normal counting mode */

    P_TMR0_IOCtrl->W = CW_TMR0_IOAMOD_Capture_PDR;
    P_TMR0_INT->W = CW_TMR0_PDCIE_Enable;      /* enable PDC0 interrupt */

    /*=====
     * Position detection control 0 setup
     *=====
     P_POS0_DectCtrl->B.PDEN = 1;           /* enable position detection */
     P_POS0_DectCtrl->B.SPLMOD = 1;         /* sample regularly */
     P_POS0_DectCtrl->B.SPLCNT = 15;
     P_POS0_DectCtrl->W |= CW_TMR0_PDCR_SPLCK_FCKdiv128;
}

```

```

} /* Init_PDC0() */

//****************************************************************************
/* IRQ1() : PDC interrupt, used for phase commutation and speed calculation */
//****************************************************************************

void IRQ1(void)      __attribute__ ((ISR));
void IRQ1(void)
{
    unsigned int n, h;

    if(P_TMR0_Status->B.PDCIF)
    {
        P_TMR0_Status->B.PDCIF = 1;
        n = P_TMR0_TGRA->W;

        if(n > P_TMR0_TCNT->W)
        {
            // continue for 6 PDR change, used for speed calculation
            // user could use P_TMR0_TGRA to achieve this.
        }

        h = P_POS0_DectData->W;           /* read current hall signal */
        /////////////////////////////////
        // do BLDC motor phase commutation
        ///////////////////////////////
    }
} /* IRQ1() */

```

【Example 11-2】: Get position detection data and read capture counter value of incoming hall sensor signal in the interrupt driven flow. Position detection change interrupt and capture module are two different and independent function peripherals. So, of course they could be both active at the same time. The captured counter value is stored to TGRA only and meanwhile issues the interrupt. The following is the sample code list.

```

#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{
    Disable_FIQ_IRQ();
    Init_TMR3 ();                      /* initialize TMR3 module */

    P_IOB_SPE->W = 0x003F;             /* enable UVW-1 output special function */
    P_TMR3_IOCctrl->W = 0x0111;        /* TIO3A ~ TIO3F output enable */
    P_TMR3_OutputCtrl->B.DUTYMODE = 0;

```

```

        /* three phase in common, use TGRA register */

P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;           /* active high */
P_POS0_DectCtrl->W = 0x0000;                                     /* sampling clock = FCK/4 */
P_POS0_DectCtrl->B.SPLMOD = 1;                                    /* sample regularly */

/* TMR0 CCP special function enable */
P_TMR0_INT->B.PDCIE = 1;                                         /* enable position change interrupt */
P_POS0_DectCtrl->B.PDEN = 1;                                       /* enable position detection */
P_POS0_DectCtrl->B.SPLMOD = 1;                                     /* sample regularly */

P_TMR0_IOCtrl->W = CW_TMR0_IOAMOD_Capture_PDR;
                                         /* input capture when PDR change */
P_TMR0_Ctrl->B.TMRPS = 1;                                         /* count on FCK/4 */
P_TMR0_Ctrl->B.CCLS = 6;                                         /* TCNT cleared by position detection change */
P_TMR0_Ctrl->B.CLEGS = 1;                                         /* TCNT cleared at rising edge */

P_TMR3_IOCtrl->W = 0x0111;
P_TMR_Start->W = CW_TMR_TMR3ST_Start; /* start TMR3 */
P_TMR_Start->W |= CW_TMR_TMR0ST_Start; /* start TMR0 */
INT_FIQ_IRQ();                                                 /* enable FIQ/IRQ channel interrupt */
}

/*****************************************/
void IRQ1(void) __attribute__ ((ISR));
void IRQ1(void)
{
    unsigned int cap_fifo;                                         /* capture value of TGRA register */
    unsigned int pdr_fifo;                                         /* position detection data */

    if(P_INT_Status->B.PDC0IF)
    {
        if(P_TMR0_Status->B.PDCIF)
        {
            P_TMR0_Status->B.PDCIF = CB_TMR0_PDCIF_Enable;
            pdr_fifo = P_POS0_DectData->W;
            cap_fifo = P_TMR0_TGRA->W;
            // do other PDR change related tasks
        }
    }
}

```

11 TPM Timer 2 Module

11.1 Introduction

SPMC75X family MCU has a general-purpose 16 bit TPM (Timer PWM Mode, TPM) timer that support functions of input capture and PWM output features. The timer 2 could be used to provide a time base system for speed loop of motor control applications. It has two timer input/output pins for input capture and PWM output operations. Figure 12-1 shows the block diagram of the timer 2 module. For details of timer 2 specifications, please refer to Table 12-1.

11.2 Timer 2 Features

- ❑ Capability to process up to two input for capture or output for PWM operation.
- ❑ Two timer general registers (P_TMR2_TGRA, P_TMR2_TGRB): assignable PWM compare match output or input capture functions.
- ❑ Two timer buffer registers (P_TMR2_TBRA, P_TMR2_TBRB) : used for PWM buffering and capture operation.
- ❑ Selection of eight programmable clock source: six internal clocks (FCK/1, FCK/4, FCK/16, FCK/64, FCK/256, FCK/1024), two external clocks (TCLKA and TCLKB).
- ❑ Programmable of timer operating modes:
 1. Timer counting modes:
 - Normal counting mode: continuous up counting.
 - PWM compare match output function: selection of 1 output, 0 output at compare match and output hold.
 - Input capture function: selection of issue capture at clock rising, falling, both edge, and position detection change event.
 2. PWM mode:
 - Two independent PWM output and can be provided with desired duty ratio.
 3. Edge-aligned PWM generation mode:
 - PWM output for normal and up counting operation.
 4. Center-aligned PWM generation mode:
 - PWM output for normal and directional up-down counting operation.
- ❑ Totally 3 interrupt sources:
 - One timer period compare match interrupt sources.
 - Timer general register P_TMR2_TGRA match interrupt sources.
 - Timer general register P_TMR2_TGRB match interrupt sources.
- ❑ Timer buffer operation:
 - The input capture register can be consisted of double buffers. The PWM compare match register can automatically be modified.

- Any initial timer compare match and period value can be set.
- Read-only 16-bit up and up/down counter register P_TMR2_TCNT register.
- R/W 16-bit timer period registers : P_TMR2_TPR provides time base of system.
- Two R/W 16-bit timer general registers : P_TMR2_TGRA, P_TMR2_TGRB are used for input capture,
- Two read-only timer buffer registers : P_TMR2_TBRA, P_TMR2_TBRB, are the buffer of above mentioned timer general registers.
- R/W 16-bit timer control register P_TMR2_Ctrl and input output control register P_TMR2_IOCtrl in which used for the selection of input capture or PWM compare match modes.
- R/W 16-bit timer interrupt enable register P_TMR2_INT provides 3 interrupt sources, and 16-bit read-only interrupt status register P_TMR2_Status in which records the interrupt flags.

Table 12-1 TPM Timer 2 Specification

Function		TPM Timer 2
Clock sources		Internal clock FCK/1, FCK/4, FCK/16, FCK/64, FCK/256, FCK/1024 External clock: TCLKA, TCLKB
IO pins		◆ TIO2A ◆ TIO2B
Timer general register		◆ P_TMR2_TGRA ◆ P_TMR2_TGRB
Timer buffer register		◆ P_TMR2_TBRA ◆ P_TMR2_TBRB
Timer period and counter register		◆ P_TMR2_TPR ◆ P_TMR2_TCNT
Capture sample clock		Internal clock: FCK/1, FCK/2, FCK/4, FCK/8
Counting edge		◆ Rising edge ◆ Falling edge ◆ Both edge
Counter clear source		◆ Cleared on P_TMR2_TGRA, P_TMR2_TGRB capture input. ◆ Cleared on P_TMR2_TPR compare matches.
Input capture function		Yes
PWM compare match output function	1 output	Yes
	0 output	Yes
	Output Hold	Yes
Edge-aligned PWM		Yes
Center-aligned PWM		Yes
Timer buffer operation		Yes
AD convert start trigger		◆ P_TMR2_TGRA compare match
Interrupt sources		◆ Timer 2 TPR interrupt ◆ Timer 2 TGRA interrupt ◆ Timer 2 TGRB interrupt

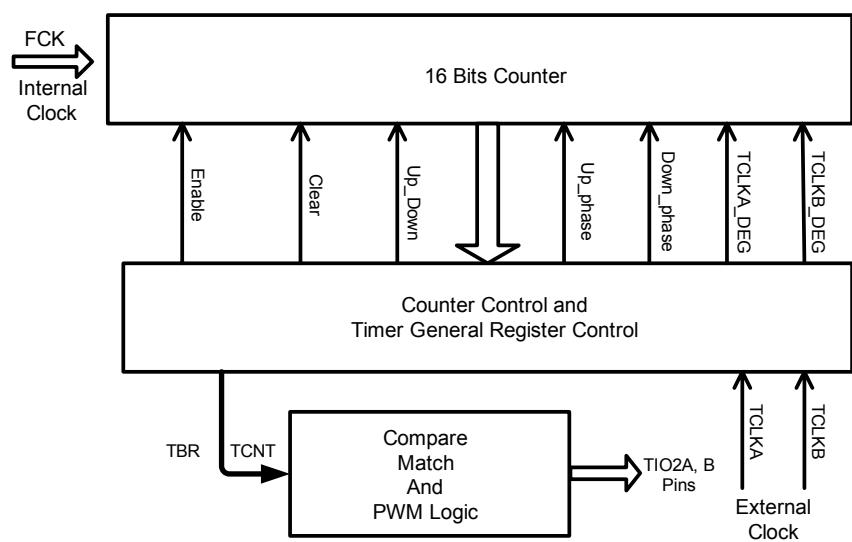


Figure 11-1 TPM timer 2 block diagram

11.3 TPM Timer 2 Input/Output/Special Function Pins

Channel	Pin name	I/O	Function
Common	TCLKA	Input	External clock A input pin
	TCLKB	Input	External clock B input pin
2	TIO2A	I/O	P_TMR2_TGRA input capture /PWM compare match output pin
	TIO2B	I/O	P_TMR2_TGRB input capture/PWM compare mathc output pin

11.4 TPM Timer 2 Operation

The TPM2 is a general-purpose timer with input capture and PWM compare match output capability. TPM timer 2 provided independent time base at different input clock sources for application such as :

The sampling and constant frequency driven features for digital control system.

Speed loop time base of inverter motor control system.

Normal operation (normal up-counting)

Count on external clock input pin TCLKA or TCLKB

Edge-aligned PWM mode (continuous up counting, PWM output mode)

Center-aligned PWM mode (continuous up/down counting, PWM output mode)

The bits value MODE in the P_TMR2_Ctrl register determines the TPM timer 2 counting mode. When the TMR2ST bit in P_TMR_Start is set to 1, the timer is starting to count from 0x0000. And the CCLS bits value in the P_TMR2_Ctrl register determines the counter clear event.

11.4.1 Continuous Up Counting Mode with Edge-Aligned PWM

The TPM2 timer channel can be configured as edge-aligned PWM mode with PWM output or normal operation without any output waveform by setting MODE bits in P_TMR2_Ctrl register.. At this mode, the timer counter act as up-counting timer and counting from 0x0000 to timer period register value. At this mode, user must set P_TMR2_TPR register and set counter clear source (CCLS) is cleared by timer period compare match.

The timer continuous up counting according to the input clock sources from bits value TMRPS defined in corresponding timer control register. The timer counter register cleared to zero until it matches that of the timer period register and period compare match event interrupt flag TPRIF is set. The period interrupt request is generated when PPRIE bit is set in P_TMR2_INT register. The general register compare match event occurs when timer counter register matches the content of TGRA or TGRB register. It generates the general register compare match interrupt when TGAIE or TGBIE bit is set in the corresponding timer interrupt enable register.

The initial value of P_TMR2_TPR can be any value from 0x0000 to 0xFFFF. Either the external clock input pin or internal clock source FCK can be selected as the clock source of the timer. The normal continuous up counting mode is extremely suitable for the generation of edge-triggered or asynchronous PWM waveforms and sampling periods in digital motor control systems. Figure 12-2 shows the normal continuous up counting mode of the TPM timer 2.

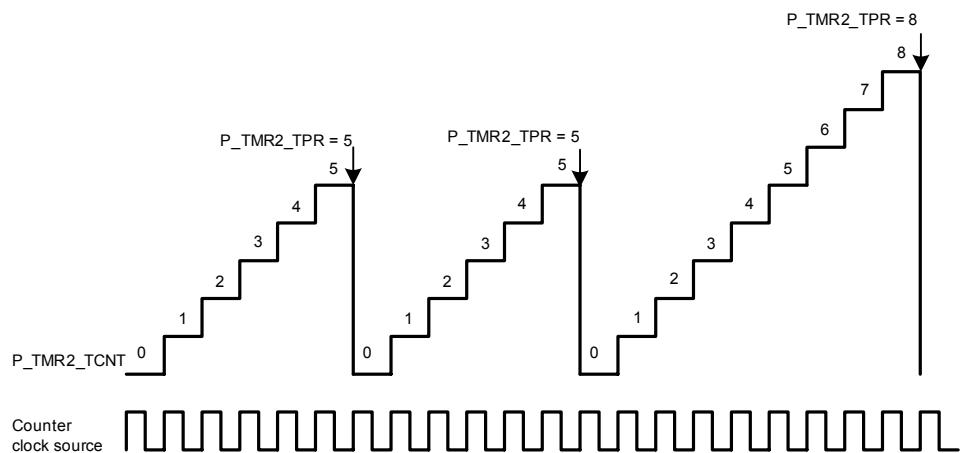
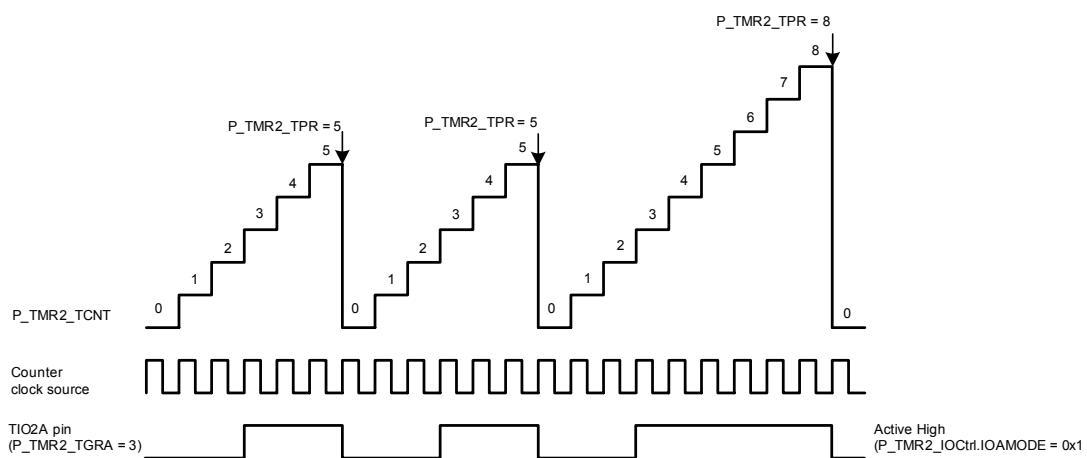
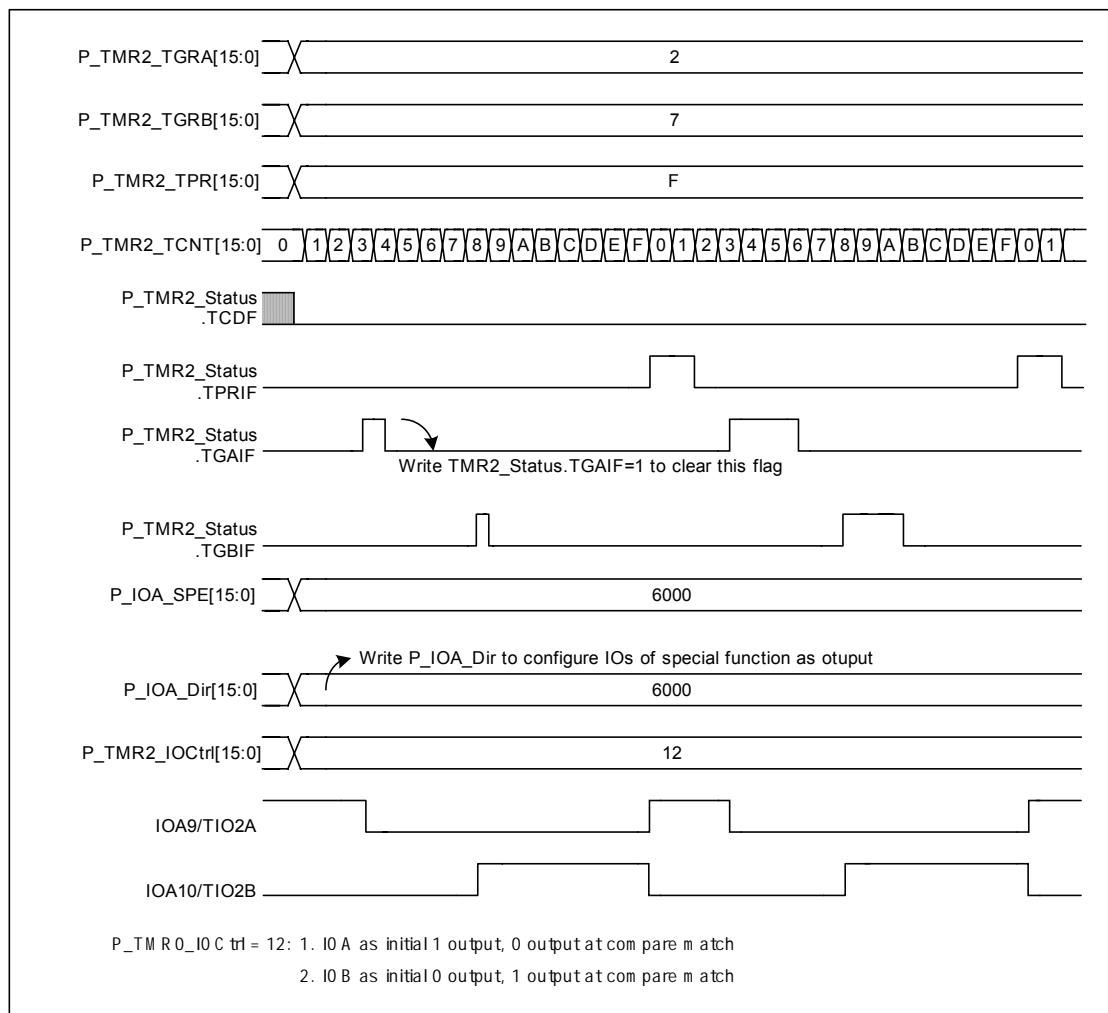


Figure 11-2 Continuous up counting example (CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

At edge-aligned PWM mode, user must set P_TMR2_TPR period register and P_TMR2_TGRx (x = A, B) general register then set counter clear source (CCLS) as cleared by timer period compare match. The compare match output condition set at P_TMR2_IOCctrl register. Figure 12-3 shows the normal continuous up counting mode for edge-aligned PWM generation of timer 2 and Figure 12-4 depicts the edge-aligned PWM timing details.


Figure 11-3 Edge-Aligned mode PWM

Figure 11-4 Edge-Aligned PWM timing of Timer 2

11.4.2 Continuous up/down counting mode with Center-Aligned PWM

The operation of continuous up/down counting mode is the same as up counting mode except the timer period register defines the middle transition point of whole counting process. The counting direction changes from up to down when the timer counter register reaches the timer period register. The period of the timer is two times of P_TMR2_TPR of the scaled clock input and the setting of CKEGS in the P_TMR2_Ctrl register. Figure 12-5 shows the continuous up/down counting mode operation..

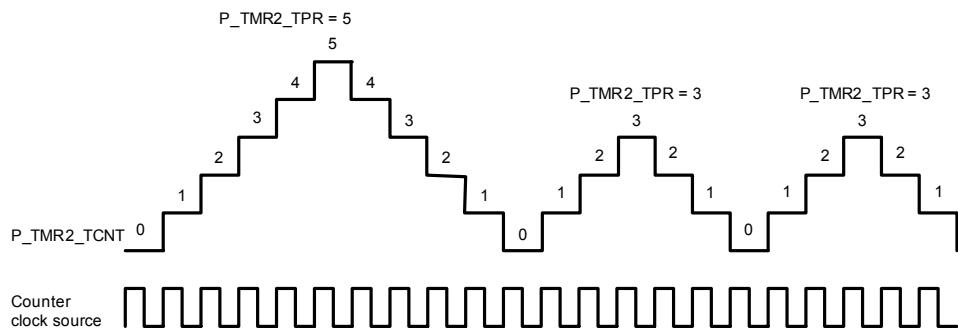


Figure 11-5 Continuous up/down counting example (CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

The initial value of the timer period register can be any value from 0x0000 to 0xFFFF. When the value of the timer counter register equals to timer period register, the TPM timer 2 start to count down to zero. The period interrupts behaves the same manner as described in the continuous up counting mode.

The counting direction is recorded at TCDF bit in the P_TMR2_Status register. Either the external clock input pin or internal clock source FCK can be selected as the clock source of the timer. Figure 12-6 shows the center -aligned mode PWM at continuous up/down counting mode of timer 2 and Figure 12-7 depicts the center-aligned PWM timing details.

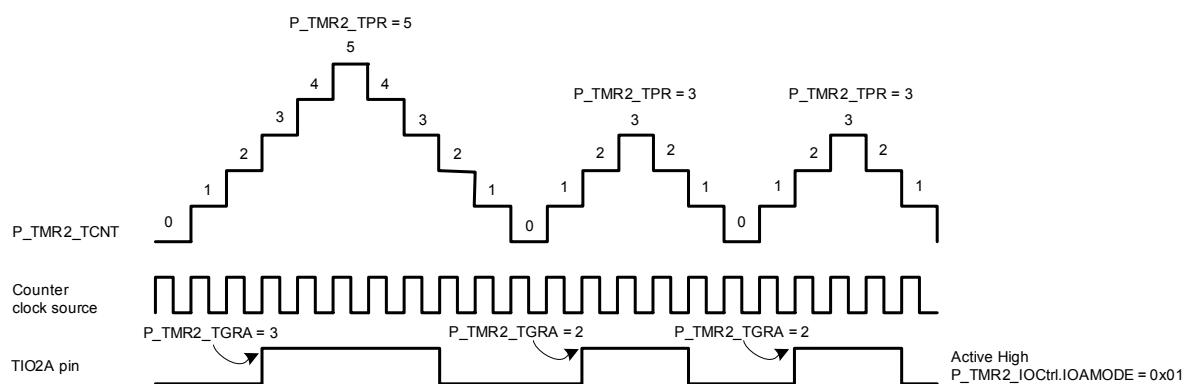


Figure 11-6 Center-Aligned mode PWM

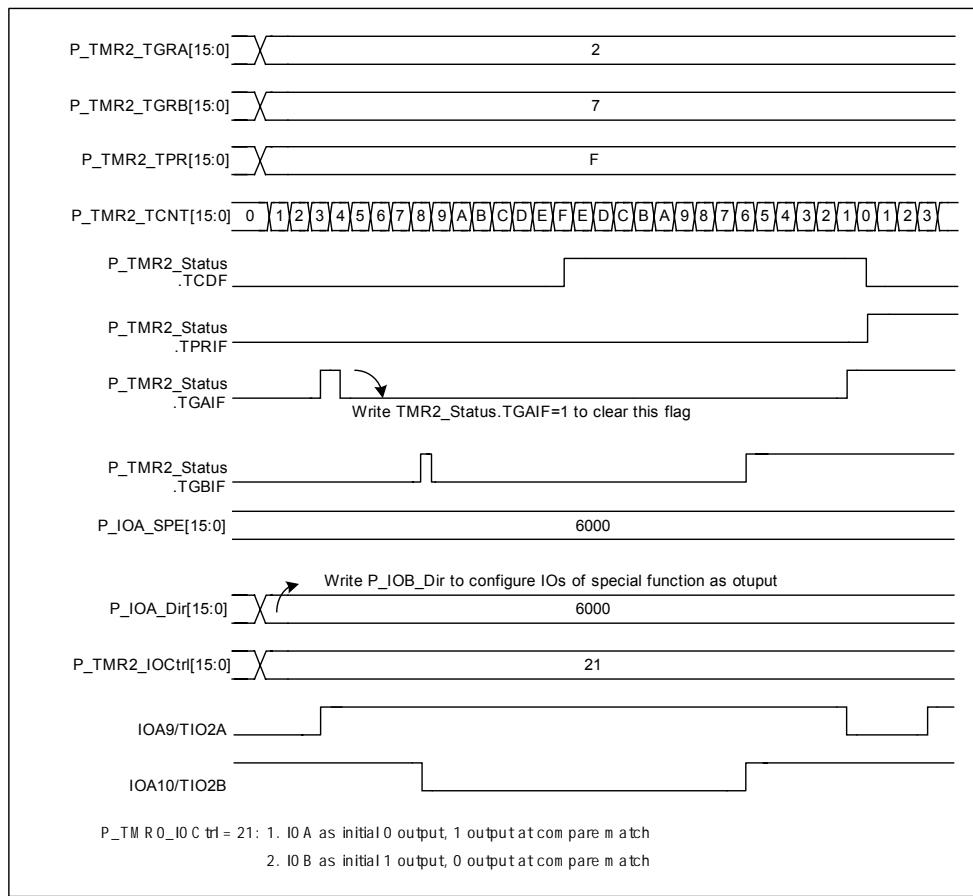


Figure 11-7 Center-Aligned PWM timing of Timer 2

11.5 Registers Descriptions

The TPM timer 2 has the following registers:

11.5.1 Common

Timer counter start register (P_TMR_Start).

11.5.2 TPM Timer 2

1. Timer 2 control register (P_TMR2_Ctrl).
2. Timer 2 input/output control register (P_TMR2_IOCctrl).
3. Timer 2 interrupt enable register (P_TMR2_INT).
4. Timer 2 interrupt status register (P_TMR2_Status).
5. Timer 2 counter register (P_TMR2_TCNT).
6. Timer 2 period register (P_TMR2_TPR).
7. Timer 2 general register A (P_TMR2_TGRA).
8. Timer 2 general register B (P_TMR2_TGRB).
9. Timer 2 buffer register A (P_TMR2_TBRA).
10. Timer 2 buffer register B (P_TMR2_TBRB).

11.6 Timer 2 Control Registers

The P_TMR2_Ctrl configures the selection of timer clock source, counter clock edge, counter clear source, counter clear edge, capture input sample clock and timer operating modes. TCLKA, TCLKB clock input will be sampled by system clock FCK. When programmed at counting on both edge, the input clock is halved. Any pulse narrower than four sampling clocks will be ignored.

11.6.1 P_TMR2_Ctrl (0x7402): Timer 2 Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK		MODE					
B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS			TMRPS	

Bit 15:14 **SPCK**: Capture input sample clock select. These bits select the capture input sample clock.

Capture input will be sampled with sample clock. Pulses shorter than four sample clocks will be considered invalid, and will be ignored.

- 00 = FCK/1
- 01 = FCK/2
- 10 = FCK/4
- 11 = FCK/8

Bit 13:10 **MODE**: Modes select. These bits are used to select the timer operation modes.

- 0xxx = Normal operation. (continuous counter up counting)
- 1x0x = Edge-aligned PWM mode. (continuous counter up counting , PWM output)
- 1x1x = Center-aligned PWM mode. (continuous counter up/down counting, PWM output)

Bit 9:8 **CLEGS**: Counter clear edge select. These bits select the counter clearing edge when the clearing source is in input capture mode.

- 00 = do not clear
- 01 = rising edge
- 10 = falling edge
- 11 = both edge

Bit 7:5 **CCLS**: Counter clear source select. These bits select the TCNT counter clearing source.

- 000 = TCNT clearing disabled
- 001 = TCNT cleared by P_TMR2_TGRA capture input
- 010 = TCNT cleared by P_TMR2_TGRB capture input
- 011 = Reserved
- 100 = Reserved

- 101 = Reserved
 110 = Reserved
 111 = TCNT cleared by P_TMR2_TPR compare match
- Bit 4:3 **CKEGS:** Clock edge select, These bits select the input clock edge. When the input clock is counted using both edges, the input clock period is halved. When FCK/1 is selected as counter clock, counter will count at rising edge if count at both edges is selected.
 00 = Count at rising edge
 01 = Count at falling edge
 1X = Count at both edges
- Bit 2:0 **TMRPS:** Timer pre-scalar select. These bits select the TCNT counter clock source. It can be selected independently for each channel.
 000 = Counts on FCK /1
 001 = Counts on FCK /4
 010 = Counts on FCK /16
 011 = Counts on FCK /64
 100 = Counts on FCK /256
 101 = Counts on FCK /1024
 110 = Counts on TCLKA pin input, maximum external clock source input is 3.0 MHz.
 111 = Counts on TCLKB pin input, maximum external clock source input is 3.0 MHz.

11.7 Timer 2 Input and Output Control Register

The P_TMR2_IOC register controls the PWM compare match output and input capture action type of TIO2A and TIO2B pins. By setting the CCLS and MODE bits in P_TMR2_Ctrl register will determine the timer IO action mode. When choosing PWM compare match output mode, the IOAMODE/IOBMODE bits determines the waveform generation depending on the active clock edge. When choosing input capture mode, the IOAMODE/IOBMODE bits defines the capture event.

11.7.1 P_TMR2_IOC (0x7412) : Timer 2 IO control register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

Bit 15:8: Reserved

Bit 7:4: **IOBMODE**: Select Timer 2 IOB Configuration

Bit 3:0: **IOAMODE**: Select Timer 2 IOA Configuration

PWM compare match output mode :

0000 = Initial output 0, 0 output at compare match

0001 = Initial output 0, 1 output at compare match

0010 = Initial output 1, 0 output at compare match

0011 = Initial output 1, 1 output at compare match

01xx = Output hold

Input capture mode :

1000 = Issue input capture interrupt at rising edge

1001 = Issue input capture interrupt at falling edge

101x = Issue input capture interrupt at both edges

11xx = Reserved

11.8 Timer 2 Interrupt Enable Register

The P_TMR2_INT register is used to enable or disable A/D conversion start request by TGRA compare match, interrupt requests for period register compare match and input capture/compare match of TGRA or TGRB.

11.8.1 P_TMR2_INT (0x7422): Timer 2 Interrupt Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							
B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	Reserved		TPRIE	Reserved		TGBIE	TGAIE

Bit 15:8 Reserved

Bit 7 **TADSE**: A/D conversion start request enable bit. Enables or disables generation of A/D conversion start request by TGRA register compare match.

0 = Disable

1 = Enable

Bit 6:5 Reserved

Bit 4 **TPRIE**: Timer Period Register interrupt enable bit. Enables or disables interrupt request by TPR register compare match.

0 = Disable

1 = Enable
 Bit 3:2 Reserved
 Bit 1 **TGBIE**: Timer General B Register interrupt enable bit. Enables or disables interrupt request by TGRB register input capture or compare match.
 0 = Disable
 1 = Enable
 Bit 0 **TGAIE**: Timer General A Register interrupt enable bit. Enables or disables interrupt request by TGRA register input capture or compare match.
 0 = Disable
 1 = Enable

11.9 Timer 2 Interrupt Status Register

The interrupt status register indicates the event generation of a period registers compare match and input capture/compare match of TGRA or TGRB. These flags show the interrupt sources. An interrupt would be generated when the corresponding interrupt enable bit is set in P_TMR2_INT register. The TCDIF represents the counter direction when timer is setup to center-aligned PWM mode.

11.9.1 P_TMR2_Status (0x7427): Timer 2 Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	Reserved	TPRIF	Reserved	Reserved	TGBIF	TGAIF	

Bit 15:8 Reserved
 Bit 7 **TCDF**: Timer Count direction flag. Status flag that shows the counting direction in which TCNT counts.
 0 = Up-counting
 1 = Down-counting
 Bit 6:5 Reserved
 Bit 4 **TPRIF**: Timer Period Register compare match flag. This status flag indicates a TPR register compare match event has been occurred. Writing ‘1’ will clear this flag.
 0 = Compare match not occurred
 1 = Compare match has occurred
 Bit 3:2 Reserved

- Bit 1 **TGBIF:** Timer General B Register input capture/compare match flag. This status flag indicates a TGRB register input capture or compare match event has been occurred. Writing '1' will clear this flag.
- 0 = Input capture/compare match not occurred
 1 = Input capture/compare match has occurred
- Bit 0 **TGAIF:** Timer General A Register input capture/compare match flag. This status flag indicates a TGRA register input capture or compare match event has been occurred. Writing '1' will clear this flag.
- 0 = Input capture/compare match not occurred
 1 = Input capture/compare match has occurred

11.10 Timer Start Register

The P_TMR_Start register selects the operation of counter start/stop for the P_TMR2_TCNT. When counter operation stopped, its contents will be cleared. Set TMR2ST to 1 would start the P_TMR2_TCNT register immediately and vice versa.

11.10.1 P_TMR_Start (0x7405): Timer Counter Start Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST
Reserved							

Bit 15:5 Reserved

Bit 4 **TMR4ST:** Timer 4 counter start setting

- 0 = Counter operation stopped
 1 = Performs counting operation

Bit 3 **TMR3ST:** Timer 3 counter start setting

- 0 = Counter operation stopped
 1 = Performs counting operation

Bit 2 **TMR2ST:** Timer 2 counter start setting

- 0 = Counter operation stopped
 1 = Performs counting operation

Bit 1 **TMR1ST:** Timer 1 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

Bit 0 **TMR0ST:** Timer 0 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

11.11 Timer 2 Counter Register

The TPM timer 2 has a 16 bit counter P_TMR2_TCNT register. It is a readable register that increments/decrements according to input clocks.

Bits TMRPS in corresponding timer control register can select input clocks. P_TMR2_TCNT increments/decrements in center-aligned PWM mode, while they only increment in other modes. The P_TMR2_TCNT register is reset to 0x0000 by compare matches with corresponding TGRA, TGRB or input captures to TGRA, TGRB.

11.11.1 P_TMR2_TCNT (0x7432): Timer 2 Counter Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

11.12 Timer 2 General and Buffer Register

TGRA, TGRB are 16-bit registers. The TPM timer 2 has two timer general registers. The TGR registers are dual function 16-bit readable/writable registers, functioning as either PWM compare match or input capture registers.

The values in TGR and TCNT are constantly compared with each other when the TGR registers are used as PWM compare match output registers. When the both values match, the TGAIF or TGBIF bit in corresponding timer interrupt status register is set to 1. Compare match outputs can be selected by TIO2A and TIO2B. When the TGR registers are used as input capture registers, the TCNT value is stored after detecting external signals. At this point, TGAIF or TGBIF bit in the corresponding timer interrupt status register is set to 1. Detection edges for input capture signals can be selected by TIO2A or TIO2B active status and programmable via CCLS bits in P_TMR2_Ctrl register.

When PWM mode, edge-aligned PWM mode, or center-aligned PWM mode is selected, the TGR register behaves as the duty ratio value register. Upon reset, the TGR registers are initialized to 0x0000.

The timer buffer registers TBRA and TBRB are the double buffers of TGRA and TGRB, respectively. The value of TGR_x (_x=A, B) can automatically be updated when the period compare match event occurs. That is, the duty ratio value will not be updated until one period ends completely. When the TBR registers are used as input capture registers, the TCNT value is stored at the falling edge of input capture port.

11.12.1 P_TMR2_TGRA (0x7446): Timer 2 General Register A

11.12.2 P_TMR2_TGRB (0x7447): Timer 2 General Register B

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

11.12.3 P_TMR2_TBRA (0x7456): Timer 2 Buffer Register A

11.12.4 P_TMR2_TBRB (0x7457): Timer 2 Buffer Register B

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

11.13 Timer 2 Period Register

The P_TMR2_TPR is a 16-bit readable/writable register. It is used to set the period of PWM waveform.

When P_TMR2_TCNT register reaches P_TMR2_TPR register value, P_TMR2_TCNT register will be cleared to 0x0000 (up-counting mode) or start down-count (continuous up-/down-counting mode) according to MODE bits programmed in P_TMR2_Ctrl register. Its default value is 0xFFFF. When P_TMR2_TPR register is set to 0x0000, the P_TMR2_TCNT register counter will stop counting and remain at 0x0000.

11.13.1 P_TMR2_TPR (0x7437): Timer 2 Period Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

11.14 TPM Timer 2 Operation

11.14.1 Normal Counting Operation

When TMR2ST is set in P_TMR_Start register, the P_TMR2_TCNT register for the corresponding channel beginning up direction counting. The counter register behaves as a free running operation and would be reset to 0x0000 when reaches 0xFFFF. Through configures the CCLS bits to 111'b, the timers is setup as periodic counter. The counter register would be reset to 0x0000 when reaches the value of P_TMR2_TPR registers. Figure 12-8 shows the programming flowchart of normal counting operation of timer 2.

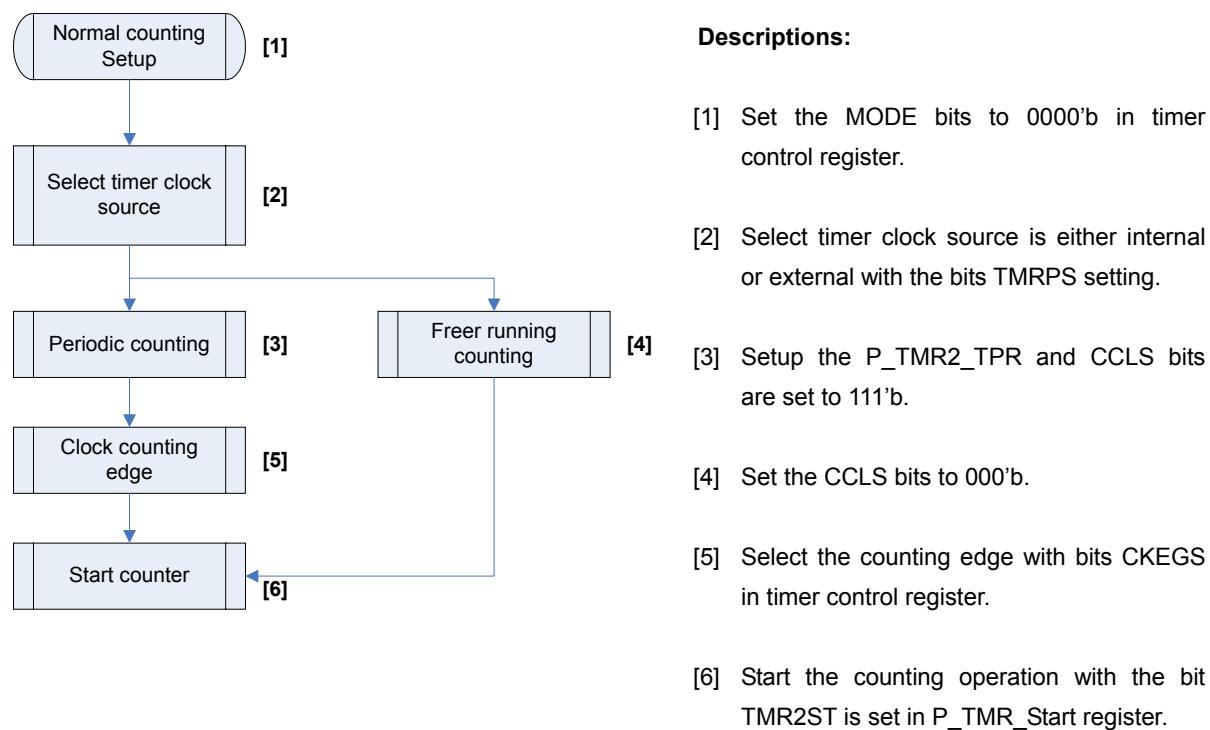


Figure 11-8 Example programming flowchart of normal counting operation

The initial value of P_TMR2_TCNT is 0x0000 and P_TMR2_TPR is 0xFFFF. When the timer start bit TMR2ST is set to 1, the P_TMR2_TCNT behaves as the free-running counter. When the counter overflows, the P_TMR2_TCNT register starts an increment operation again from 0x0000.

When a period compare match event is selected as P_TMR2_TCNT register clearing source, the counter behaves as the periodic counting operation. The P_TMR2_TPR register for setting the period the period and counter-clearing source is selected by setting CCLS bits to 111'b. After the settings have made, the counter register start an increment operation as periodic counter when timer start bit TMR2ST is set to 1. When counter matches the value of P_TMR2_TPR register, the TPRIF flag is set to 1 in timer interrupt status register and counter register is reset to 0x0000. If the bit value of TPRIE in P_TMR2_INT register is set to 1, the timer 2 requests an interrupt.

11.14.2 PWM Compare Match Output Operation

The TPM timer 2 module can perform PWM compare match output function up to two pins output. The output waveforms have active low at compare match, active high at compare match and output hold for the corresponding TIO2A and TIO2B output pin using compare match with P_TMR2_TGRA and P_TMR2_TGRB register respectively. Figure 12-9 shows the programming flowchart of PWM compare match output operation. In Figure 12-10 shows the compare match output and interrupt timing.

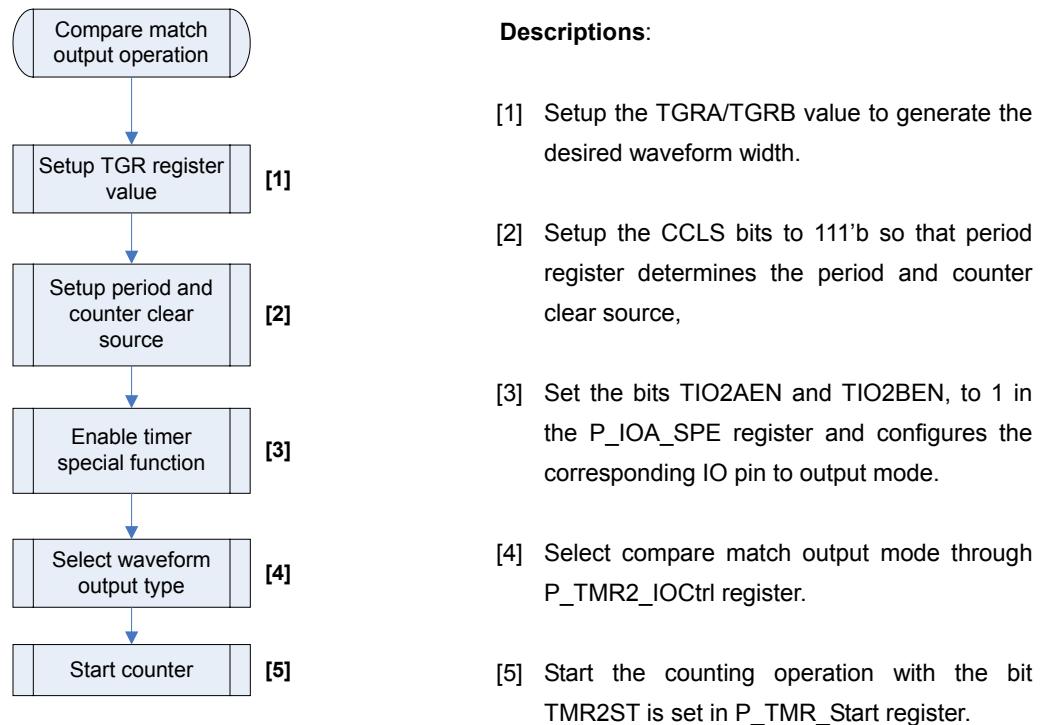


Figure 11-9 Example programming flowchart of PWM compare match output operation

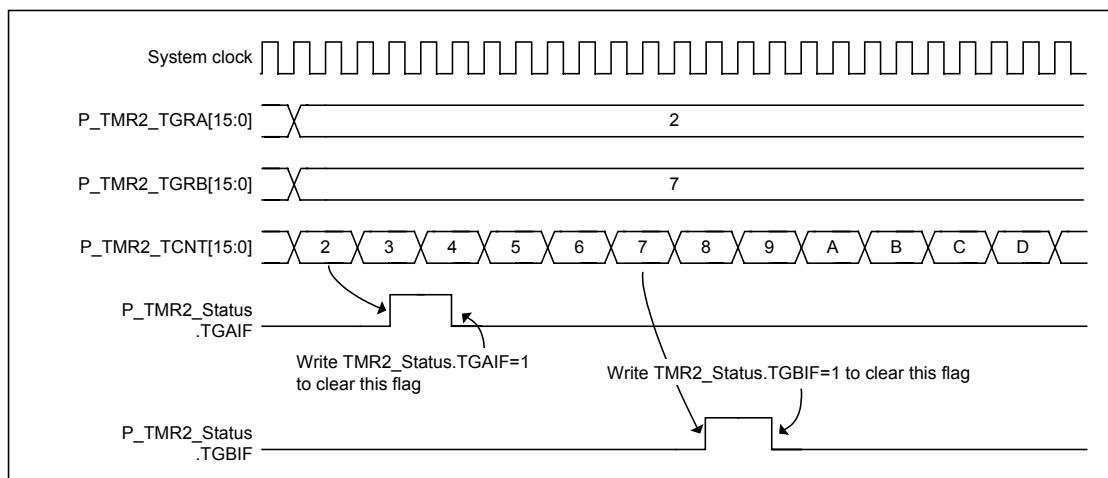


Figure 11-10 compare match output and interrupt timing

11.14.3 Input Capture Operation

The capture function activation and the input edge at which the interrupt status flag issued are determined by the bits of IOAMOD and IOBMOD in the P_TMR2_IOCrl register, respectively. It can be the rising edge, falling edge or both edge. The value of counter is always transferred to TGRx ($x=A, B$) and TBRx ($x=A, B$) at the rising and falling edge of corresponding input capture port, respectively. The counter register, P_TMR2_TCNT can be cleared according to the setting of CCLS in P_TMR2_Ctrl register. The counter clear source can be one of TIO0A and TIO0B at the selected edge according to CLEGS in P_TMR2_Ctrl.

Table 12-2 shows the input capture configurations settings and results. When the input capture function is selected, the pulse width or period can be measured presents on input pin. Figure 12-13 shows the programming flowchart of input capture operation. Figure 12-12 is an example of input capture TIO0x ($x=A, B$). The correlations between the configuration of P_TMR2_IOCrl and interrupt even are shown.

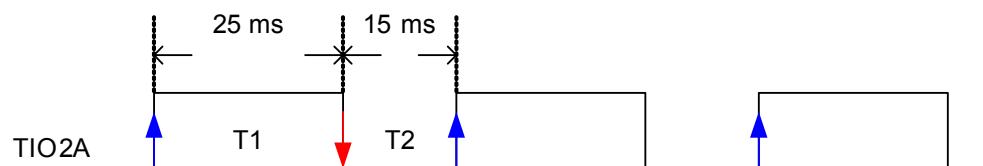
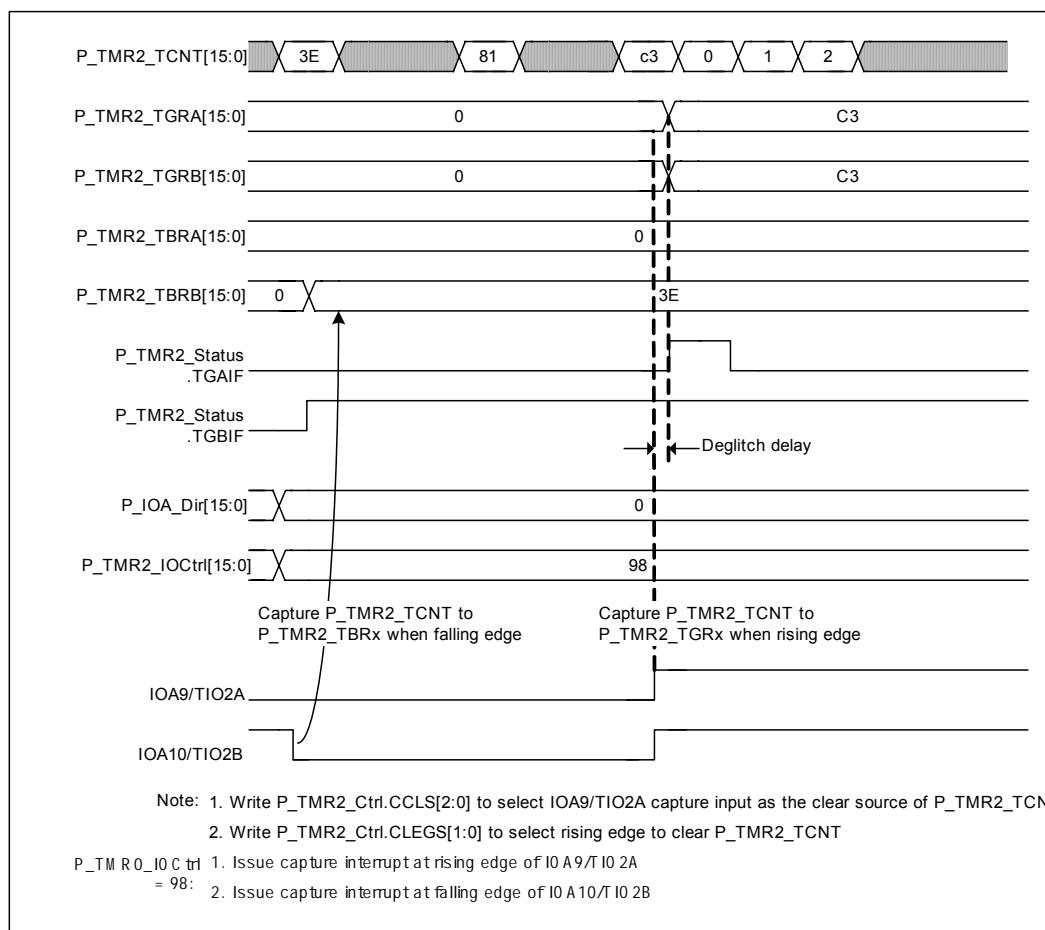


Figure 11-11 input capture signal connected to TIO2A

Table 12-2 input capture configuration settings and results

Input capture settings			Enter interrupt at signal edge	Results
CLEGS	CCLS	IOAMODE		
Rising	TGRA	Rising	Rising edge	P_TMR2_TCNT = period (40 ms)

Input capture settings			Enter interrupt at signal edge	Results
CLEGS	CCLS	IOAMODE		
Edge		edge		P_TMR2_TBRA = T1 (25ms)
Rising Edge	TGRA	Falling edge	Falling edge	P_TMR2_TGRA = period (40 ms) P_TMR2_TBRA = T1 (25ms)
Rising Edge	TGRA	Both edge	Both edge	P_TMR2_TGRA = period (40 ms) P_TMR2_TBRA = T1 (25ms)
Falling Edge	TGRA	Rising edge	Rising edge	P_TMR2_TGRA = T2 (15 ms) P_TMR2_TBRA = period (40ms)
Falling Edge	TGRA	Falling edge	Falling edge	P_TMR2_TGRA = T2 (15 ms) P_TMR2_TBRA = period (40ms)
Falling Edge	TGRA	Both edge	Both edge	P_TMR2_TGRA = T2 (15 ms) P_TMR2_TBRA = period (40ms)
Both Edge	TGRA	Rising edge	Rising edge	P_TMR2_TGRA = T2 (15 ms) P_TMR2_TBRA = T1 (25ms)
Both Edge	TGRA	Falling edge	Falling edge	P_TMR2_TGRA = T2 (15 ms) P_TMR2_TBRA = T1 (25ms)
Both Edge	TGRA	Both edge	Both edge	P_TMR2_TGRA = T2 (15 ms) P_TMR2_TBRA = T1 (25ms)


Figure 11-12 Capture input signal pulse width and pulse cycle

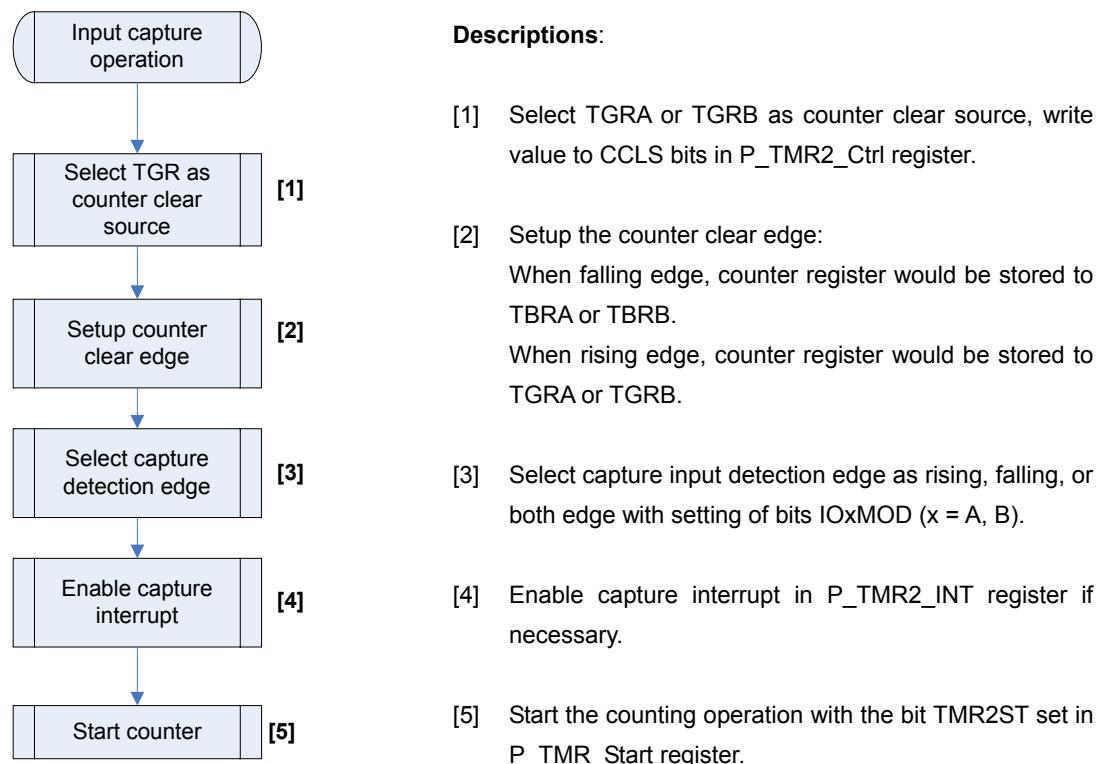


Figure 11-13 Example programming flowchart of input capture operation

11.15 Application Example Design Tips

【Example 12-1】 : The following shows the example setting for generation the period interrupt of timer 2 and compare match interrupt of P_TMR2_TGRA and P_TMR2_TGRB.

```

#include "Spmc75_regs.h"
#include "Spmc_typedef.h"

void Init_TMR2(void)
{
    P_TMR2_Ctrl->W = CW_TMR2_CKEGS_Rising | CW_TMR2_CCLS_TPR;
    P_TMR2_Ctrl->W |= CW_TMR2_MODE_PWM_Edge;

    P_TMR2_TPR->W = 24000000 / 4000; /* PWM carrier freq. = 4000 Hz */
    P_TMR2_Ctrl->W |= CW_TMR2_TMRPS_FCKdiv1;
    P_IOA_SPE->B.TIO2AEN = 1;
    P_IOA_SPE->B.TIO2BEN = 1;

    P_IOA_Dir->B.bit9 = 1;
    P_IOA_Dir->B.bit10 = 1;

    P_TMR2_IOCctrl->W = CW_TMR2_IOAMOD_Output_01 | CW_TMR2_IOBMOD_Output_01;
    P_TMR2_INT->W = CW_TMR2_TPRIE_Enable;
    P_TMR2_INT->W |= (CW_TMR2_TGAIE_Enable | CW_TMR2_TGBIE_Enable);
}
  
```

```
P_TMR_Start->B.TMR2ST = 1;
} /* Init_TMR2() */

/*********************************************
/* IRQ4() : IRQ4 interrupt service routine, possibile int. sources is      */
/*          TPM2IF                                                       */
/*********************************************
void IRQ4(void)    __attribute__ ((ISR));
void IRQ4(void)
{
    if(P_TMR2_Status->B.TPRIF)
    {
        P_TMR2_Status->B.TPRIF = 1;
        TPM2_TPRINT_ISR();
    }

    if(P_TMR2_Status->B.TGAIF)
    {
        P_TMR2_Status->B.TGAIF = 1;
        TPM2_TGRAINT_ISR();
    }

    if(P_TMR2_Status->B.TGBIF)
    {
        P_TMR2_Status->B.TGBIF = 1;
        TPM2_TGRBINT_ISR();
    }
} /* IRQ4() */

/*********************************************
/* TPM2_TPRINT_ISR() : TPM2 period interrupt           */
/*********************************************
void TPM2_TPRINT_ISR(void)
{
    P_TMR2_TGRA->W = (5 * P_TMR2_TPR->W) / 100;                      /* 5 % duty */
    P_TMR2_TGRB->W = (95L * P_TMR2_TPR->W) / 100;                     /* 95 % duty */
} /* TPM2_TPRINT_ISR() */

/*********************************************
/* TPM2_TGRAINT_ISR() : TPM2 general register A compare match interrupt   */
/*********************************************
void TPM2_TGRAINT_ISR(void)
{
    P_TMR2_TGRA->W = (5 * P_TMR2_TPR->W) / 100;                      /* 5 % duty */
```

```
    } /* TPM2_TGRAINT_ISR */  
  
/* ***** */  
/* TPM2_TGRBINT_ISR() : TPM2 general register B compare match interrupt */  
/* ***** */  
void TPM2_TGRBINT_ISR(void)  
{  
    P_TMR2_TGRB->W = (95L * P_TMR2_TPR->W) / 100;           /* 95 % duty */  
} /* TPM2_TGRBINT_ISR */
```

12 MCP Timer 3 and 4 Module

12.1 Introduction

There are two channels of 16bit MCP(Motor Control PWM) timers, MCP timer 3 and MCP timer 4 on the SPMC75X family MCU chip. The MCP timers provide two independent set of full function for three-phase, six programmable PWM waveform output capability. The MCP timer 3 should work with PDC timer 0 and MCP timer 4 works with PDC timer 1 to form the speed closed loop control for BLDC and ACI motor applications. This MCP timer module has totally twelve timer output pins for motor control operations. Figure 13-1 shows the block diagram of the MCP timer 3 and 4 module. For details of timer specifications, please refer to Table 13-1.

12.2 MCP Timer 3 and 4 Features

- ❑ Capability to generate up to twelve programmable PWM waveform.
- ❑ Six timer general registers (TGRAx/TGRBx/TGRCx, x = 3, 4): three registers for each channel independently assignable PWM compare match output functions.
- ❑ Six timer buffer registers (TBRAx/TBRBx/TBRCx, x = 3, 4) : three registers for each channel used for PWM buffering operation.
- ❑ Partial load of duty value of PWM generation problem prevention: provide a load control register to make the duty value for PWM could be load simultaneously.
- ❑ Selection of eight programmable clock source: six internal clocks (FCK/1, FCK/4, FCK/16, FCK/64, FCK/256, FCK/1024), two external clocks (TCLKA and TCLKB).
- ❑ Programmable of timer operating modes:
 1. Timer counting modes:
 - Normal counting mode: continuous up counting.
 - PWM compare match output function: selection of 1 output, 0 output at compare match and output hold.
 2. PWM mode:
 - Two independent set of six-phase PWM output can be provided with desired duty ratio.
 3. Edge-aligned PWM generation mode:
 - PWM output for normal and up counting operation.
 4. Center-aligned PWM generation mode:
- ❑ Totally 10 interrupt sources, five for each channel:
 - Timer period compare match interrupt.
 - TGRD register compare match interrupt sources.
 - External fault input interrupt.
 - External overload input interrupt.

- PWM output short protection interrupt.
- UVW phases output synchronization source select: synchronized to position data register P_POSx_DectData ($x = 0, 1$) change, TGRB or TGRC register compare match.
- PWM duty mode selection: use TGRA or three phase are independent.
- Timer buffer operation:
 - The PWM compare match register can automatically be modified.
- Any initial timer compare match and period value can be set.
- Read-only 16-bit up and up/down counter register P_TMRx_TCNT ($x = 3, 4$) register.
- R/W 16-bit timer period registers : P_TMRx_TPR ($x = 3, 4$) provides time base of system.
- Four R/W 16-bit timer general registers : P_TMRx_TGRA, P_TMRx_TGRB, P_TMRx_TGRC, P_TMRx_TGRD ($x = 3, 4$), are used for PWM compare match output.
- Three read-only timer buffer registers : P_TMRx_TBRA, P_TMRx_TBRB, P_TMRx_TBRC ($x = 3, 4$), are the buffer of above mentioned timer general registers.
- R/W 16-bit timer control register P_TMRx_Ctrl and input output control register P_TMRx_IOCtrl ($x = 3, 4$) in which used for the selection of PWM compare match output modes.
- R/W 16-bit timer output control register P_TMRx_OutputCtrl ($x = 3, 4$), determines the PWM generation modes.
- R/W 16-bit timer interrupt enable register P_TMRx_INT provides 2 interrupt sources, and 16-bit R/W interrupt status register P_TMRx_Status ($x = 3, 4$) in which records the interrupt flags.
- R/W 16-bit dead-time setting and control register P_TMRx_DeadTime ($x = 3, 4$).
- R/W 16-bit input fault control P_Faultx_Ctrl ($x = 1, 2$), fault release P_Faultx_Release ($x = 1, 2$) and load protection P_OLx_Ctrl ($x = 1, 2$) registers.

Table 13-1 MCP timer 3 and 4 specification

Function	MCP Timer 3	MCP Timer 4
Clock sources	Internal clock FCK/1, FCK/4, FCK/16, FCK/64, FCK/256, FCK/1024,	
	External clock: TCLKA, TCLKB	
Output pins	♦ TIO3A/U1 ♦ TIO3B/V1 ♦ TIO3C/W1 ♦ TIO3D/U1N ♦ TIO3E/V1N ♦ TIO3F/W1N	♦ TIO4A/U2 ♦ TIO4B/V2 ♦ TIO4C/W2 ♦ TIO4D/U2N ♦ TIO4E/V2N ♦ TIO4F/W2N
Timer general Register	♦ P_TMR3_TGRA, P_TMR3_TGRB, ♦ P_TMR3_TGRC, P_TMR3_TGRD	♦ P_TMR4_TGRA, P_TMR4_TGRB, ♦ P_TMR4_TGRC, P_TMR4_TGRD
Timer buffer Register	♦ P_TMR3_TBRA, P_TMR3_TBRB, ♦ P_TMR3_TBRC	♦ P_TMR4_TBRA, P_TMR4_TBRB, ♦ P_TMR4_TBRC
Interrupt period	Interrupt every one, two, four and eight period	

Function		MCP Timer 3	MCP Timer 4
Counting edge		<ul style="list-style-type: none"> ◆ Rising edge ◆ Falling edge ◆ Both edge 	
Counter clear source		Clear on P_TMR3_TPR compare match	Clear on P_TMR4_TPR compare match
PWM compare match output function	1 output	Yes	Yes
	0 output	Yes	Yes
	Output Hold	Yes	Yes
BLDC motor drive PWM		Yes	Yes
ACI motor drive PWM		Yes	Yes
Edge-aligned PWM		Yes	Yes
Center-aligned PWM		Yes	Yes
Complementary PWM		Yes	Yes
Timer buffer operation		Yes, but not P_TMR3_TGRD	Yes, but not P_TMR4_TGRD
AD convert start trigger		P_TMR3_TGRD compare match	P_TMR4_TGRD compare match
PWM duty partial load prevention		Yes, through P_TMR_LDOK register	Yes, through P_TMR_LDOK register
PWM output enable control		Yes, through P_TMR_Output register	Yes, through P_TMR_Output register
PWM waveform control	Forced H	Yes	Yes
	Forced L	Yes	Yes
	Active H	Yes	Yes
	Active L	Yes	Yes
UVW phase synchronization		<ul style="list-style-type: none"> ◆ P_POS0_DectData register change ◆ P_TMR3_TGRB compare match ◆ P_TMR3_TGRC compare match 	<ul style="list-style-type: none"> ◆ P_POS1_DectData register change ◆ P_TMR4_TGRB compare match ◆ P_TMR4_TGRC compare match
Duty Mode		Use P_TMR3_TGRA register or thee timer general register	Use P_TMR4_TGRA register or thee timer general register
MCP registers write protection		Yes, through P_TPWM_Write register	Yes, through P_TPWM_Write Register
External fault input pin		FTIN1	FTIN2
External overload input pin		OL1	OL2

Function	MCP Timer 3	MCP Timer 4
Interrupt source	<ul style="list-style-type: none"> ◆ Timer 3 TPR interrupt ◆ Timer 3 TGRA interrupt ◆ External fault input 1 interrupt ◆ External overload input 1 interrupt ◆ MCP timer 3 PWM output short interrupt 	<ul style="list-style-type: none"> ◆ Timer 4 TPR interrupt ◆ Timer 4 TGRA interrupt ◆ MCP4 external fault input 2 interrupt ◆ External overload input 2 interrupt ◆ MCP timer 4 PWM output short interrupt

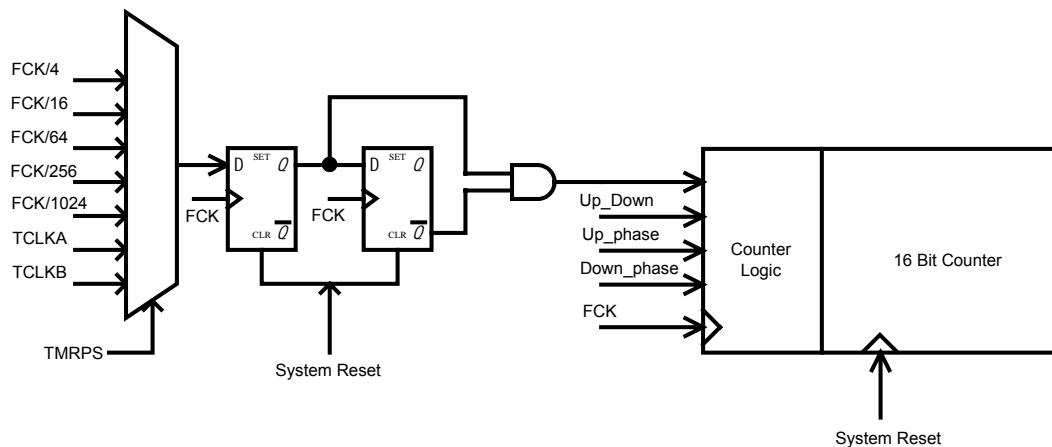


Figure 12-1 MCP timer 3 and 4 block diagram

12.3 MCP Timer 3 and 4 Input/Output/Special Function Pins

Channel	Pin name	I/O	Function
Common	TCLKA	Input	External clock A input pin
	TCLKB	Input	External clock B input pin
3	TIO3A	Output	P_TMR3_TGRA PWM compare match output pin
	TIO3B	Output	P_TMR3_TGRB PWM compare match output pin
	TIO3C	Output	P_TMR3_TGRC PWM compare match output pin
	TIO3D	Output	Programmable through P_TMR3_OutputCtrl register and TIO3A state
	TIO3E	Output	Programmable through P_TMR3_OutputCtrl register and TIO3B state
	TIO3F	Output	Programmable through P_TMR3_OutputCtrl register and TIO3C state
	FTIN1	Input	External fault input pin 1
	OL1	Input	External overload input pin 1
4	TIO4A	Output	P_TMR4_TGRA PWM compare match output pin
	TIO4B	Output	P_TMR4_TGRB PWM compare match output pin
	TIO4C	Output	P_TMR4_TGRC PWM compare match output pin
	TIO4D	Output	Programmable through P_TMR4_OutputCtrl register and TIO4A state

Channel	Pin name	I/O	Function
	TIO4E	Output	Programmable through P_TMR4_OutputCtrl register and TIO4B state
	TIO4F	Output	Programmable through P_TMR4_OutputCtrl register and TIO4C state
	FTIN2	Input	External fault input pin 2
	OL2	Input	External overload input pin 2

12.4 MCP Timer 3 and 4 Counting Operation

The on-chip MCP timer 3 and 4 have the following five possible counting operations :

Normal operation (normal up-counting).

Count on external clock input pin TCLKA or TCLKB.

Edge-aligned PWM mode (continuous up counting, PWM output mode).

Center-aligned PWM mode (continuous up/down counting, PWM output mode).

Complementary PWM mode w/o dead-time control.

The bits value MODE in the P_TMRx_Ctrl ($x = 3, 4$) register determines the MCP timer counting mode.

When the corresponding timer enable bit in P_TMR_Start is set to 1, the timer is starting to count from 0x0000. Also the CCLS bits value in the P_TMRx_Ctrl ($x = 3, 4$) register determines the counter clear event and PRDINT bits value select the P_TMRx_TPR ($x = 3, 4$) interrupt frequency.

12.4.1 Continuous Up Counting Mode with Edge-Aligned PWM

The MCP timers can be configured as edge-aligned PWM mode with PWM output or normal operation without any output waveform by setting MODE bits in P_TMRx_Ctrl ($x = 3, 4$) register. At this mode, the timer counter act as up-counting timer and counting from 0x0000 to timer period register value. At this mode, user must set P_TMRx_TPR ($x = 3 \sim 4$) register and set counter clear source (CCLS) as cleared by timer period compare match and also needs to setup proper bits value of PRDINT in the P_TMRx_Ctrl ($x = 3, 4$) register.

The MCP timer continuous up counting according to the input clock sources from bits value TMRPS defined in corresponding timer control register. The timer counter register cleared to zero until it matches that of the timer period register and period compare match event interrupt flag TPRIF is set. The period interrupt request is generated when PPRIE bit is set in P_TMRx_INT ($x = 3, 4$) register.

The general register compare match event occurs when timer counter register matches the content of P_TMRx_TGRD ($x = 3, 4$) register and this event could trigger an ADC to start a conversion.

The initial value of P_TMRx_TPR ($x = 3, 4$) can be any value from 0x0000 to 0xFFFF. Either the external clock input pin or internal clock source FCK can be selected as the clock source of the timer. The normal

continuous up counting mode is extremely suitable for the generation of edge-triggered or asynchronous PWM waveforms and sampling periods in digital motor control systems. Figure 13-2 shows the normal continuous up counting mode of the MCP timer 3.

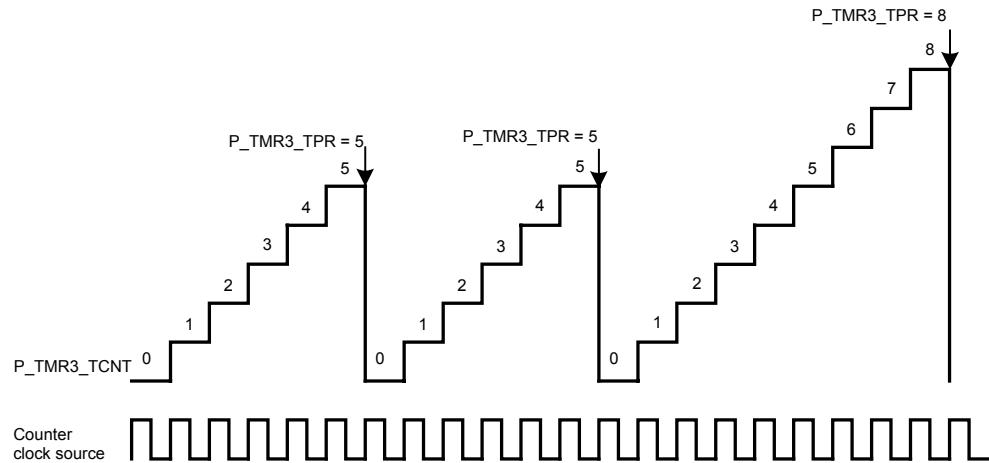


Figure 12-2 Continuous up counting example (CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

At edge-aligned PWM mode, user must set P_TMRx_TPR ($x = 3, 4$) period register and P_TMRx_TGRy ($y = A, B, C$) general register then set counter clear source (CCLS) as cleared by timer period compare match. The compare match output condition set at P_TMRx_IOCctrl ($x = 3, 4$) register. Figure 13-3 shows the normal continuous up counting mode for edge-aligned PWM generation of timer 3.

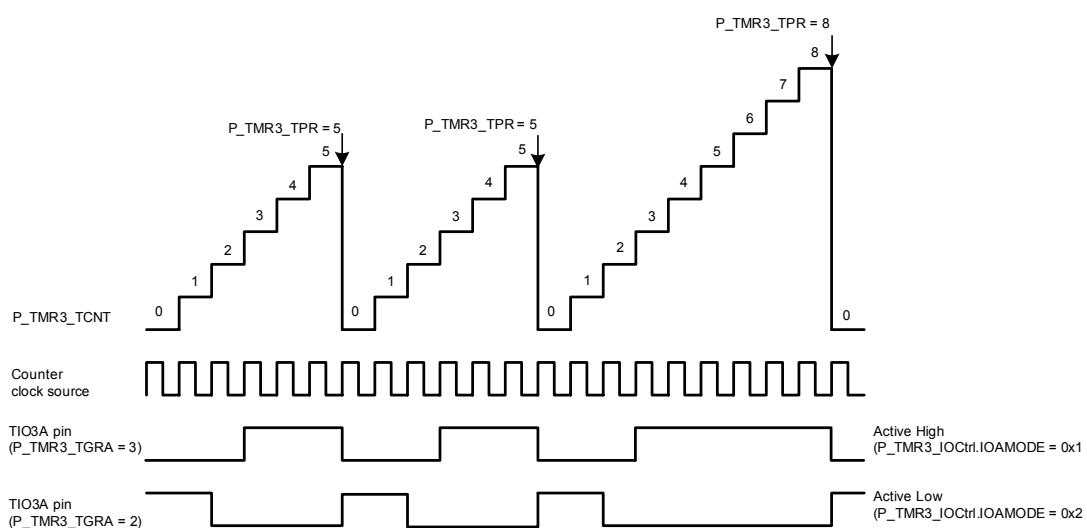


Figure 12-3 Edge-Aligned mode PWM

12.4.2 Continuous up/down counting mode with Center-Aligned PWM

The operation of continuous up/down counting mode is the same as up counting mode except the timer period register defines the middle transition point of whole counting process. The counting direction changes from up to down when the timer counter register reaches the timer period register. The period of the timer is two times of P_TMRx_TPR ($x = 3, 4$) of the scaled clock input and the setting of CKEGS in the P_TMRx_Ctrl ($x = 3, 4$) register. Figure 13-4 shows the continuous up/down counting mode operation.

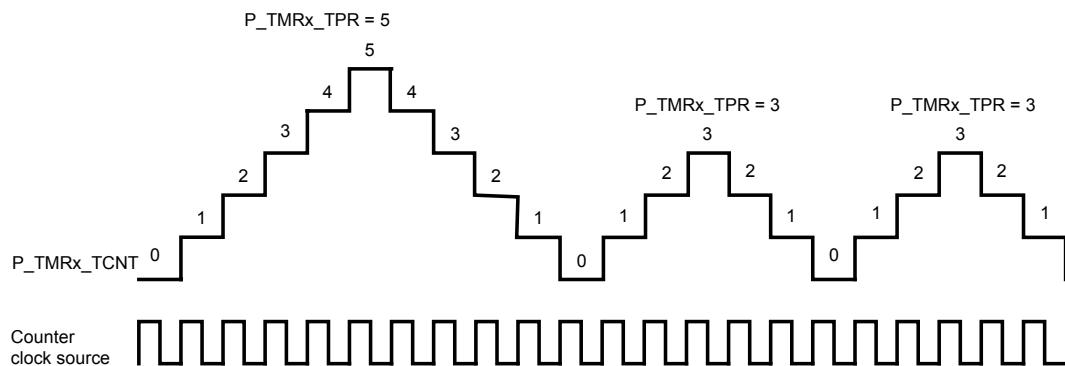


Figure 12-4 Continuous up/down counting example (CCLS = 111'b, CKEGS = 00'b, TMRPS = 000'b)

The initial value of the timer period register can be any value from 0x0000 to 0xFFFF. When the value of the timer counter register equals to timer period register, the MCP timer start to count down to zero. The period interrupt behaves the same manner as described in the continuous up counting mode.

The counting direction is recorded at TCDF bit in the P_TMRx_Status ($x = 3, 4$) register. Either the external clock input pin or internal clock source FCK can be selected as the clock source of the timer. Figure 13-5 shows the center -aligned mode PWM at continuous up/down counting mode of timer 3.

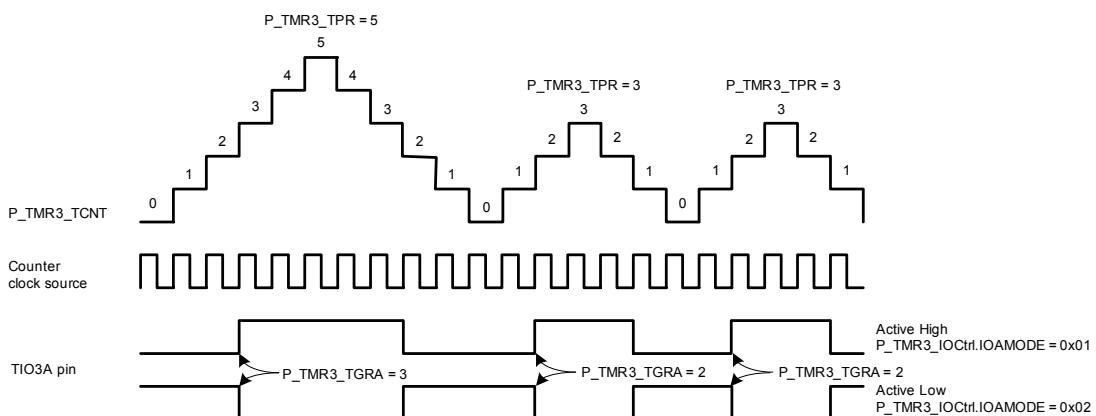


Figure 12-5 Center-Aligned mode PWM

12.5 Registers Descriptions

The MCP timer 3 and 4 have the following registers:

12.5.1 Common

1. Timer counter start register (P_TMR_Start).
2. Timer output enable register (P_TMR_Output).
3. Timer/PWM module write enable control register (P_TPWM_Write).
4. Timer Load-OK register (P_TMR_LDOK).

12.5.2 MCP Timer 3

1. Timer 3 control register (P_TMR3_Ctrl).
2. Timer 3 input/output control register (P_TMR3_IOCtrl).
3. Timer 3 interrupt enable register (P_TMR3_INT).
4. Timer 3 interrupt status register (P_TMR3_Status).
5. Timer 3 output control register (P_TMR3_OutputCtrl).
6. Timer 3 dead time and control register (P_TMR3_DeadTime).
7. Fault input 1 control and status register (P_Fault1_Ctrl).
8. Fault 1 flag release register (P_Fault1_Release).
9. Overload input 1 control and status register (P_OL1_Ctrl).
10. Timer 3 counter register (P_TMR3_TCNT).
11. Timer 3 period register (P_TMR3_TPR).
12. Timer 3 general register A (P_TMR3_TGRA).
13. Timer 3 general register B (P_TMR3_TGRB).
14. Timer 3 general register C (P_TMR3_TGRC).
15. Timer 3 general register D (P_TMR3_TGRD).
16. Timer 3 buffer register A (P_TMR3_TBRA).
17. Timer 3 buffer register B (P_TMR3_TBRB).
18. Timer 3 buffer register C (P_TMR3_TBRB).

12.5.3 MCP Timer 4

1. Timer 4 control register (P_TMR4_Ctrl).
2. Timer 4 input/output control register (P_TMR4_IOCtrl).
3. Timer 4 interrupt enable register (P_TMR4_INT).
4. Timer 4 interrupt status register (P_TMR4_Status).
5. Timer 4 output control register (P_TMR4_OutputCtrl).
6. Timer 4 dead time and control register (P_TMR4_DeadTime).
7. Fault input 2 control and status register (P_Fault2_Ctrl).

8. Fault 2 flag release register (P_Fault2_Release).
9. Overload input 2 control and status register (P_OL2_Ctrl).
10. Timer 4 counter register (P_TMR4_TCNT).
11. Timer 4 period register (P_TMR4_TPR).
12. Timer 4 general register A (P_TMR4_TGRA).
13. Timer 4 general register B (P_TMR4_TGRB).
14. Timer 4 general register C (P_TMR4_TGRC).
15. Timer 4 general register D (P_TMR4_TGRD).
16. Timer 4 buffer register A (P_TMR4_TBRA).
17. Timer 4 buffer register B (P_TMR4_TBRB).
18. Timer 4 buffer register C (P_TMR4_TBRB).

12.6 Timer 3 and 4 Control Registers

The P_TMRx_Ctrl ($x = 3, 4$) configures the selection of timer clock source, counter clock edge, counter clear source, TPR interrupt frequency and timer operating modes. TCLKA, TCLKB clock input will be sampled by system clock FCK. Any pulse narrower than four sampling clocks will be ignored. The MCP timer 3 and 4 does not support input capture mode. When programmed at counting on both edge, the input clock is halved.

12.6.1 P_TMR3_Ctrl (0x7403) : Timer 3 Control Register

12.6.2 P_TMR4_Ctrl (0x7404) : Timer 4 Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PRDINT		MODE				Reserved	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS			CKEGS			TMRPS	

Bit 15:14 **PRDINT**: TPR interrupt frequency select. These bits select the TPR interrupt frequency.

- 00 = Interrupt every period
- 01 = Interrupt once every 2 periods
- 10 = Interrupt once every 4 periods
- 11 = Interrupt once every 8 periods

Bit 13:10 **MODE**: Modes select. These bits are used to select the timer operation modes.

- 0xxx = Normal operation. (continuous counter up counting)
- 1x0x = Edge-aligned PWM mode. (continuous counter up counting, PWM output)

- 1x1x = Center-aligned PWM mode. (continuous counter up/down counting, PWM output)
- Bit 9:8 Reserved
- Bit 7:5 **CCLS:** Counter clear source select. These bits select the TCNT ($x = 3, 4$) counter clearing source.
- 000 = TCNT clearing disabled
 - 001 = Reserved
 - 010 = Reserved
 - 011 = Reserved
 - 100 = Reserved
 - 101 = Reserved
 - 110 = Reserved
 - 111 = TCNT cleared by P_TMRx_TPR ($x = 3, 4$) compare match
- Bit 4:3 **CKEWS:** Clock edge select, These bits select the input clock edge. When the input clock is counted using both edges, the input clock period is halved. When FCK/1 is selected as counter clock, counter will count at rising edge if count at both edges is selected.
- 00 = Count at rising edge
 - 01 = Count at falling edge
 - 1X = Count at both edges
- Bit 2:0 **TMRPS:** Timer pre-scalar select. These bits select the TCNT counter clock source. It can be selected independently for each channel.
- 000 = Counts on FCK /1
 - 001 = Counts on FCK /4
 - 010 = Counts on FCK /16
 - 011 = Counts on FCK /64
 - 100 = Counts on FCK /256
 - 101 = Counts on FCK /1024
 - 110 = Counts on TCLKA pin input, maximum external clock source input is 3.0 MHz.
 - 111 = Counts on TCLKB pin input, maximum external clock source input is 3.0 MHz.

12.7 Timer 3 and 4 Input and Output Control Register

The P_TMRx_IOCrl ($x = 3, 4$) register controls the PWM compare match output action type of TIOxA, TIOxB, and TIOxC ($x = 3, 4$) pins. By setting the CCLS and MODE bits in P_TMRx_Ctrl ($x = 3, 4$) register will determine the timer action mode. When choosing PWM compare match output mode, the IOAMODE/IOBMODE/IOCMODE bits determines the waveform generation depending on the active clock edge. The MCP 3 and 4 does not have the setting for input capture operation and bits value 1xxx'b of IOAMODE/IOBMODE/IOCMODE are invalid.

12.7.1 P_TMR3_IOCtrl (0x7413) : Timer 3 IO control register

12.7.2 P_TMR4_IOCtrl (0x7414) : Timer 4 IO control register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved				IOCMODE			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

Bit 15:12: Reserved

Bit 11:8: **IOCMODE**: Select Timer 3/Timer 4 IOC Configuration

Bit 7:4: **IOBMODE**: Select Timer 3/Timer 4 IOB Configuration

Bit 3:0: **IOAMODE**: Select Timer 3/Timer 4 IOA Configuration

PWM compare match output mode:

0000 = Initial output 0, 0 output at compare match

0001 = Initial output 0, 1 output at compare match

0010 = Initial output 1, 0 output at compare match

0011 = Initial output 1, 1 output at compare match

01xx = Output hold

1xxx = Reserved

12.8 Timer 3 and 4 Interrupt Enable Register

The P_TMRx_INT (x = 3, 4) register is used to enable or disable A/D conversion start request by TGRD compare match, interrupt requests for period register compare match and TGRD register compare mach.

12.8.1 MCP timer 3 channel interrupt sources

The timer 3 has two interrupt sources : TPRIF and TGDF. The TPRIF interrupt is generated when the P_TMR3_TCNT counter match the P_TMR3_TPR register with TPRIE bit set in P_TMR3_INT register. The P_TMR3_TPR register determines the period of the timer 3. The TPRIF interrupt flag can be masked by clearing TPRIE bit in P_TMR3_INT register. When the TPRIF bit sets and enters the interrupt service routine, the TPRIF bit must be cleared by writing '1' to TPRIF bit in P_TMR3_Status register before re-enabling the interrupt.

The TGDF interrupt is issued when P_TMR3_TGRD register content match the P_TMR3_TCNT register with TGDFE bit set in P_TMR3_INT register. The TGDF interrupt flag can be masked by clearing TGDFE bit in P_TMR3_INT register. When the TGDF bit sets and enters the interrupt service routine, the TGDF

bit must be cleared by writing ‘1’ before re-enabling the interrupt.

The timer 3 interrupt can be two-way entries. IRQ3 is the default entry, and FIQ by setting the MCP3IP bit in P_INT_Priority register.

12.8.2 MCP timer 4 channel interrupt sources

The timer 4 has two interrupt sources : TPRIF and TGDF. The TPRIF interrupt is generated when the P_TMR4_TCNT counter match the P_TMR4_TPR register with TPRIE bit set in P_TMR4_INT register. The P_TMR4_TPR register determines the period of the timer 4. The TPRIF interrupt flag can be masked by clearing TPRIE bit in P_TMR4_INT register. When the TPRIF bit sets and enters the interrupt service routine, the TPRIF bit must be cleared by writing ‘1’ to TPRIF bit in P_TMR4_Status register before re-enabling the interrupt.

The TGDF interrupt is issued when P_TMR4_TGRD register content match the P_TMR4_TCNT register with TGDFE bit set in P_TMR4_INT register. The TGDF interrupt flag can be masked by clearing TGDFE bit in P_TMR4_INT register. When the TGDF bit sets and enters the interrupt service routine, the TGDF bit must be cleared by writing ‘1’ before re-enabling the interrupt.

The timer 4 interrupt can be two-way entries. IRQ3 is the default entry, and FIQ by setting the MCP4IP bit in P_INT_Priority register.

12.8.3 P_TMR3_INT (0x7423): Timer 3 Interrupt Enable Register

12.8.4 P_TMR4_INT (0x7424): Timer 4 Interrupt Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TADSE	Reserved		TPRIE	TGDFE	Reserved		

Bit 15:8 Reserved

Bit 7 **TADSE:** A/D conversion start request enable bit. Enables or disables generation of A/D conversion start request by TGRD register compare match.

0 = Disable

1 = Enable

Bit 6:5 Reserved

Bit 4 **TPRIE:** Timer Period Register interrupt enable bit. Enables or disables interrupt request by TPR register compare match.

0 = Disable

1 = Enable

- Bit 3 **TGDIE:** Timer General D Register interrupt enable bit. Enables or disables interrupt request by TGRD register compare match.
- 0 = Disable
- 1 = Enable
- Bit 2:0 Reserved

12.9 Timer 3 and 4 Interrupt Status Register

The interrupt status register indicates the event generation of period register compare match and compare match of TGRD. These flags show the interrupt sources. An interrupt would be generated when the corresponding interrupt enable bit is set in P_TMRx_INT (x = 3, 4) register.

The TCDF represents the counter direction when timer is setup to center-aligned PWM mode.

12.9.1 P_TMR3_Status (0x7428): Timer 3 Interrupt Status Register

12.9.2 P_TMR4_Status (0x7429): Timer 4 Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TCDF	Reserved		TPRIF	TGdif	Reserved		

Bit 15:8 Reserved

- Bit 7 **TCDF:** Timer Count direction flag. Status flag that shows the counting direction in which TCNT counts.

0 = Up-counting

1 = Down-counting

Bit 6:5 Reserved

- Bit 4 **TPRIF:** Timer Period Register compare match flag. This status flag indicates a TPR register compare match event has been occurred. Writing '1' will clear this flag.

0 = Compare match not occurred

1 = Compare match has occurred

- Bit 3 **TGDIF:** Timer General D Register compare match flag. This status flag indicates a TGRD register compare match event has been occurred. Writing '1' will clear this flag.
- 0 = Compare match not occurred
 1 = Compare match has occurred
- Bit 2:0 Reserved

12.10 Timer Output Enable Register

This register enables/disables the PWM outputs of the specified MCP3 and/or MCP4 timer module. The PWM output will be high-impedance if disabled. Note that this register only takes effect when TIO3A to TIO3F or TIO4A to TIO4F are set to be output pins in special function mode.

12.10.1 P_TMR_Output (0x7406): Timer Output Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved	TMR4FOE	TMR4EOE	TMR4DOE	TMR4COE	TMR4BOE	TMR4AOE	

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved	TMR3FOE	TMR3EOE	TMR3DOE	TMR3COE	TMR3BOE	TMR3AOE	

Bit 15:14 Reserved

- Bit 13 **TMR4FOE:** Timer 4 IOF Output enable (TIO4F). This bit enables/disables the TIO4F pin output.

 0 = Disable
 1 = Enable

- Bit 12 **TMR4EOE:** Timer 4 IOE Output enable (TIO4E). This bit enables/disables the TIO4E pin output.

 0 = Disable
 1 = Enable

- Bit 11 **TMR4DOE:** Timer 4 IOD Output enable (TIO4D). This bit enables/disables the TIO4D pin output.

 0 = Disable
 1 = Enable

- Bit 10 **TMR4COE:** Timer 4 IOC Output enable (TIO4C). This bit enables/disables the TIO4C pin output.

 0 = Disable
 1 = Enable

- Bit 9 **TMR4BOE:** Timer 4 IOB Output enable (TIO4B). This bit enables/disables the TIO4B pin output.
0 = Disable
1 = Enable
- Bit 8 **TMR4AOE:** Timer 4 IOA Output enable (TIO4A). This bit enables/disables the TIO4A pin output.
0 = Disable
1 = Enable
- Bit 7:6 Reserved
- Bit 5 **TMR3FOE:** Timer 3 IOF Output enable (TIO3F). This bit enables/disables the TIO3F pin output.
0 = Disable
1 = Enable
- Bit 4 **TMR3EOE:** Timer 3 IOE Output enable (TIO3E). This bit enables/disables the TIO3E pin output.
0 = Disable
1 = Enable
- Bit 3 **TMR3DOE:** Timer 3 IOD Output enable (TIO3D). This bit enables/disables the TIO3D pin output.
0 = Disable
1: Enable
- Bit 2 **TMR3COE:** Timer 3 IOC Output enable (TIO3C). This bit enables/disables the TIO3C pin output.
0 = Disable
1 = Enable
- Bit 1 **TMR3BOE:** Timer 3 IOB Output enable (TIO3B). This bit enables/disables the TIO3B pin output.
0 = Disable
1 = Enable
- Bit 0 **TMR3AOE:** Timer 3 IOA Output enable (TIO3A). This bit enables/disables the TIO3A pin output.
0 = Disable
1 = Enable

12.11 Timer 3 and 4 Output Control Register

The MCP timer 3 and 4 output control register value is an important setting for the PWM waveform type used for motor drive applications. The DUTYMODE bit determines the duty ratio register used for PWM. Generally speaking, when driving a BLDC motor with 120 degree PWM mode only P_TMRx_TGRA (x = 3, 4) is need for setup the duty register. In other words, all three P_TMRx_TGRA / P_TMRx_TGRB / P_TMRx_TGRC (x = 3, 4) registers required for 180 degree PWM, including BLDC and ACI motor and the corresponding timing diagram is shown in Figure 13-6.

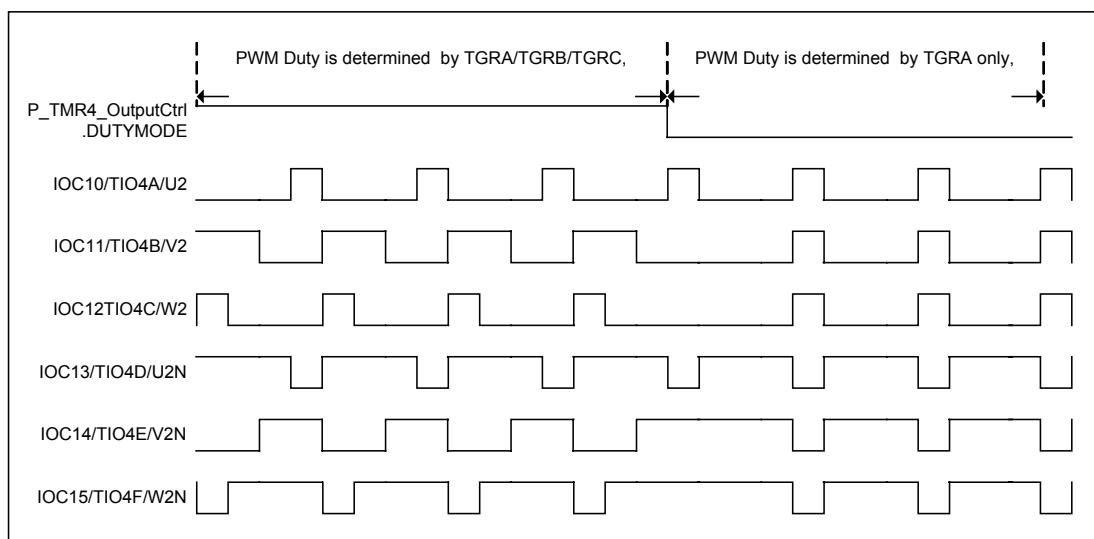


Figure 12-6 PWM output timing with different DUTYMODE bit settings

The POLP bit determines the PWM active level for IGBT/MOSFET switching device. The UPWM, VPWM and WPWM can be forced H/L or active H/L waveform on specified pin. Bits POLP, PWM/VWPM/UPWM and WOC/VOC/UOC result in different kinds of PWM waveform generation. Table 13-2 and table 13-3 show the output polarity and waveform control of U phase; the same manner could apply to V phase and W phase. Figure 13-7 indicates the PWM output polarity timing.

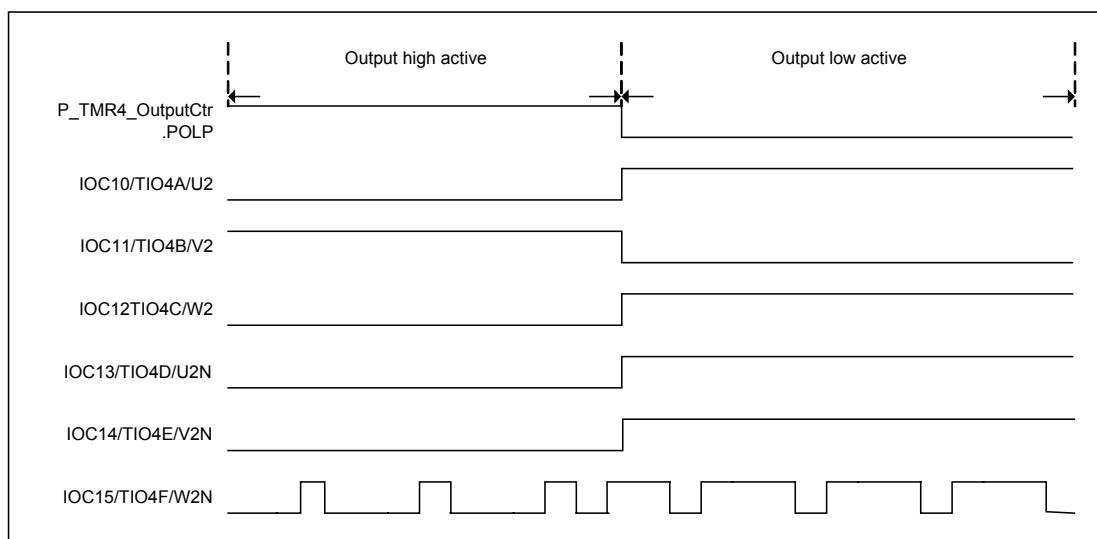


Figure 12-7 PWM output polarity timing

12.11.1 P_TMR3_OutputCtrl (0x7407): Timer 3 Output Control Register

12.11.2 P_TMR4_OutputCtrl (0x7408): Timer 4 Output Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
DUTYMODE	POLP	Reserved			WPWM	VPWM	UPWM

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SYNC		WOC		VOC		UOC	

Bit 15 **DUTYMODE:** Duty mode select. This bit determines which registers used for PWM duty ratio control.

0 = U phase in common (same as TGRA register)

1 = Three phases independent

Bit 14 **POLP:** Upper phase polarity select. This bit selects the upper phase polarity.

0 = Active low

1 = Active high

Bit 13:11 Reserved

Bit 10 **WPWM:** W phase PWM output select

0 = H/L level output

1 = PWM waveform output

- Bit 9 **VPWM:** V phase PWM output select
0 = H/L level output.
1 = PWM waveform output
- Bit 8 **UPWM:** U phase PWM output select
0 = H/L level output
1 = PWM waveform output
- Bit 7:6 **SYNC:** UVW phases output synchronization source select.
00 = No sync
01 = Synchronized to P_POSx_DectData (x = 0, 1) register change
10 = Synchronized to TGRB register compare match
11 = Synchronized to TGRC register compare match
- Bit 5:4 **WOC:** W phase output control
- Bit 3::2 **VOC:** V phase output control
- Bit 1:0 **UOC:** U phase output control

In Table 13-2 and 13-3 shows the PWM waveform combination according the bits value POLP and UOC/VOC/WOC in the P_TMRx_OutputCtrl (x = 3, 4) register.

There are four PWM waveform output mode Mode0 ~ Mode3 defined in P_TMRx_OutputCtrl (x = 3, 4) registers. According to bits [10:8] and bits [7:0] settings, we can generate different waveform combinations on Ux/UxN, Vx/VxN and Wx/WxN (x= 1, 2) output pin pairs. Note that the settings of POLP bit and WPWM, VPWM, UPWM bits determine the waveform is active related to P_TMRx_TCNT (x = 3, 4) counter clock source or simply forced output logic high or low immediately.

Mode0 and Mode3 are used for complementary PWM generation. Typical application example is to drive AC induction motor in which utilize sine-wave PWM methodology. Also these modes could be applied to 180-degree BLDC inverter drive application. When use Mode0 or Mode3 for complementary PWM generation, the DTP bits should be set the desired dead-time to protect the driving circuit, and DTWE, DTVE, DTUE should be set to 1 to enable dead-time asserted on the specified phase according to developer's application.

Mode1 and Mode2 are normal PWM mode, the dead-time feature is disable in the related phase.

Table 13-2 U phase output polarity: Active high (POLP=1)

UOC[1,0]			UPWM			
			1: PWM output		0: H/L output	
			U phase	UN phase	U phase	UN phase
Mode 0	0	0	CPWM	PWM	L	L
Mode 1	0	1	L	PWM	L	H
Mode 2	1	0	PWM	L	H	L
Mode 3	1	1	PWM	CPWM	H	H

Table 13-3 U phase output polarity: Active low (POLP=0)

UOC[1,0]			UPWM			
			1: PWM output		0: H/L output	
			U phase	UN phase	U phase	UN phase
Mode0	0	0	PWM	CPWM	H	H
Mode1	0	1	H	CPWM	H	L
Mode2	1	0	CPWM	H	L	H
Mode3	1	1	CPWM	PWM	L	L

The SYNC bit determines the UVW PWM phases that can be synchronized with different internal events sources. Possible synchronization sources are P_POSx_Data (x = 0, 1) register changes event, P_TMRx_TGRB (x = 0, 1) compare match event and P_TMRx_TGRC (x = 0, 1) compare match event. The Figure 13-8 shows the PWM synchronization with different sources.

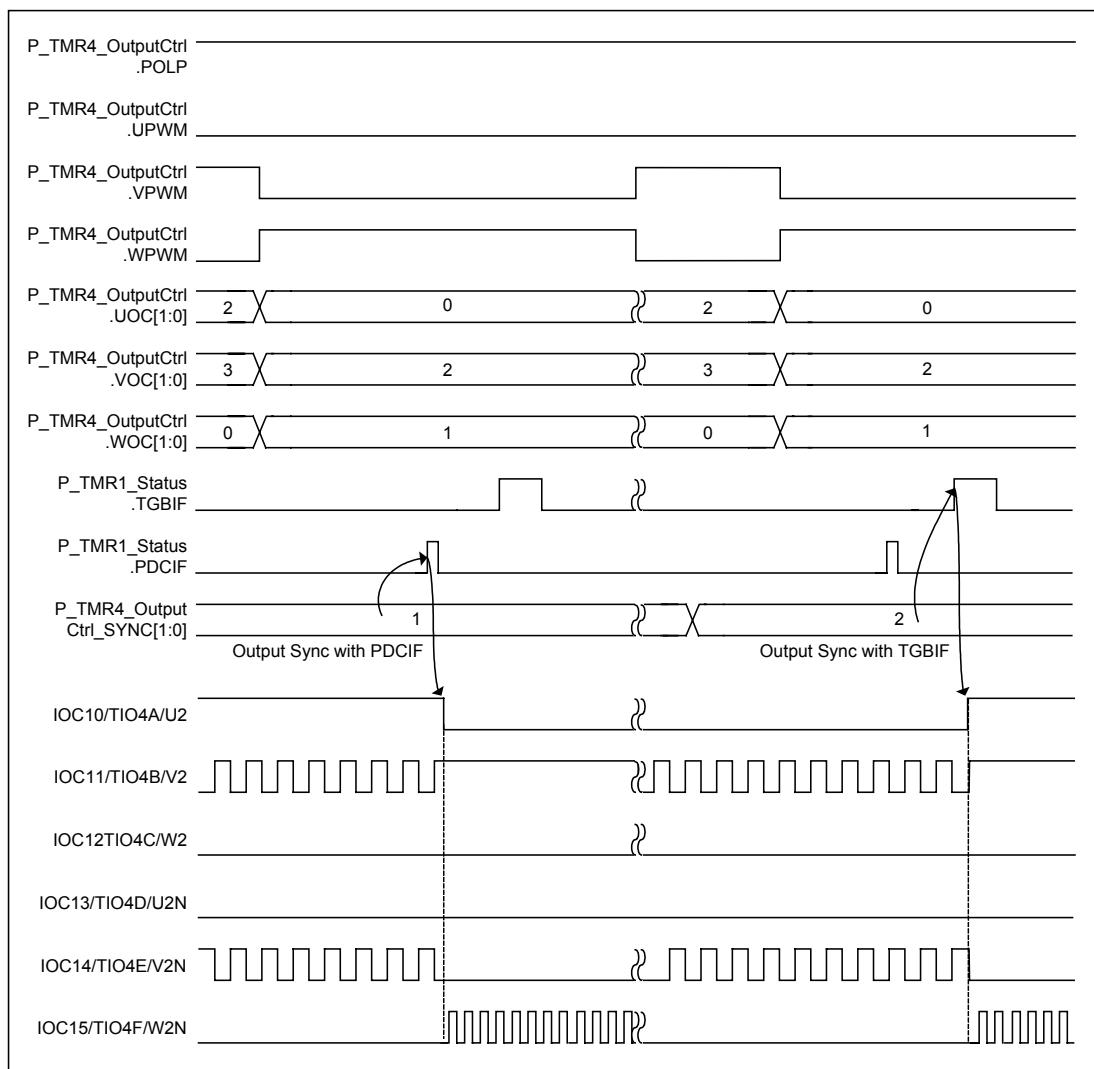


Figure 12-8 PWM synchronization mode

12.12 Timer 3 and 4 Dead Time and Control Register

In complementary PWM mode, each complementary PWM channels pairs can be used to drive the high side and low side transistors. The PWM signal of each pairs should be totally logic opposite in non-dead case, but actually condition is not. To prevent the active time of PWM signal between low side and high side PWM are overlapping, the dead time unit must be used in complementary PWM mode. Figure 13-9 shows the center-aligned complementary PWM with dead time inserted of timer 4.

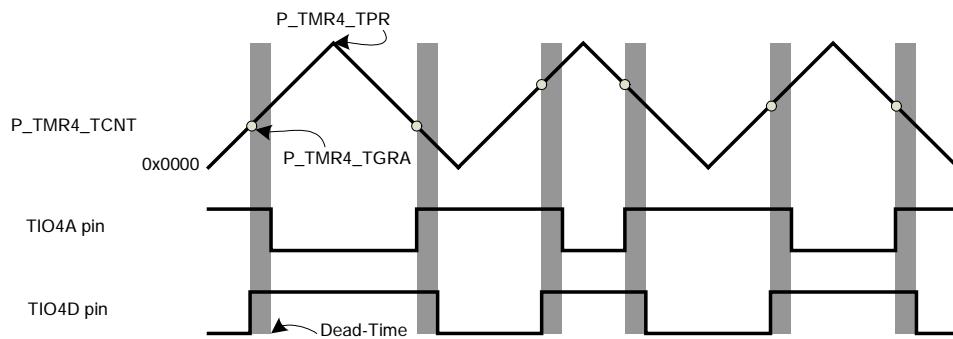


Figure 12-9 Active-low PWM mode of dead-time generation

There are two dead-time timer control registers in the SPMC75X family MCU : P_TMR3_DeadTime and P_TMR4_DeadTime, for MCP channel 3 and channel 4. The dead-time timer only works when programmed in complementary PWM mode. The dead-time timer unit will delay the active edge for positive or lower phase output and affected by the setting of POLP bit in the P_TMRx_OutputCtrl ($x = 3, 4$) register. Three phase dead time feature could be independent enable or disable and the dead time interval is determined through DTP bits at FCK/4 clock source.

12.12.1 P_TMR3_DeadTime (0x7460): Timer 3 Dead Time and Control Register

12.12.2 P_TMR4_DeadTime (0x7461): Timer 4 Dead Time and Control Register

B15	B14	B13	B12	B11	B10	B9	B8		
R	R/W	RW	R/W	R	R	R	R		
0	0	0	0	0	0	0	0		
Reserved	DTWE	DTVE	DTUE	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W						
0	0	0	0	0	0	0	0
Reserved	DTP						

Bit 15 Reserved

Bit 14 **DTWE**: Dead-time timer enable for W phases

0 = Disable

1 = Enable

Bit 13 **DTVE**: Dead-time timer enable for V phases

0 = Disable

1 = Enable

Bit 12 **DTUE**: Dead-time timer enable for U phases

0 = Disable

1 = Enable

Bit 11:7 Reserved

Bit 6:0 **DTP:** Dead-time timer period. These bits select the dead-time period. Dead time can be set from 0 to 127 FCK/4 clocks.

12.13 Timer Fault Input Control Register

The fault protection input can be used to establish a high-impedance state for protection by applying an active low state on FTINT1-2 pins summarized in table 13-4. Also, an interrupt will be generated simultaneously. The PWM outputs will remain in high-impedance state until released. Refer to Figure 13-10 to see the fault error on IOC9/FTIN2 timing details.

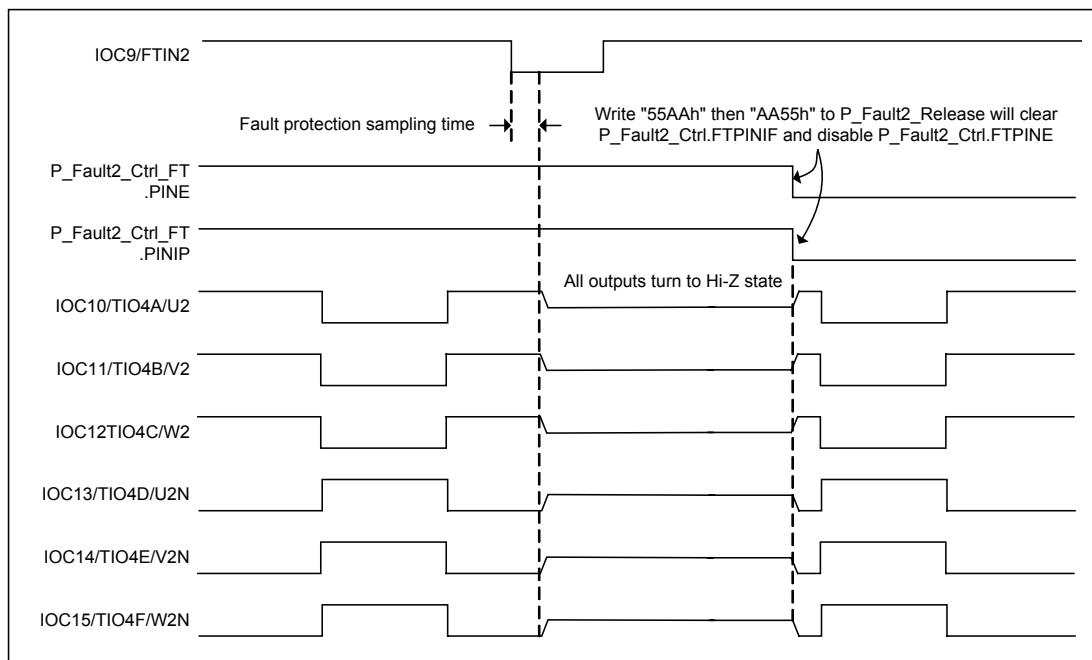


Figure 12-10 Fault input error timing

Also note that the OCLS bit in P_Faultx_Ctrl ($x = 1, 2$) register determines the PWM output compare polarity level; developer must properly select the PWM protection level to ensure the safety of the driver circuits of target system. User should aware that the fault input protection only works with complementary PWM mode. Figure 13-11 indicates the output compare error timing.

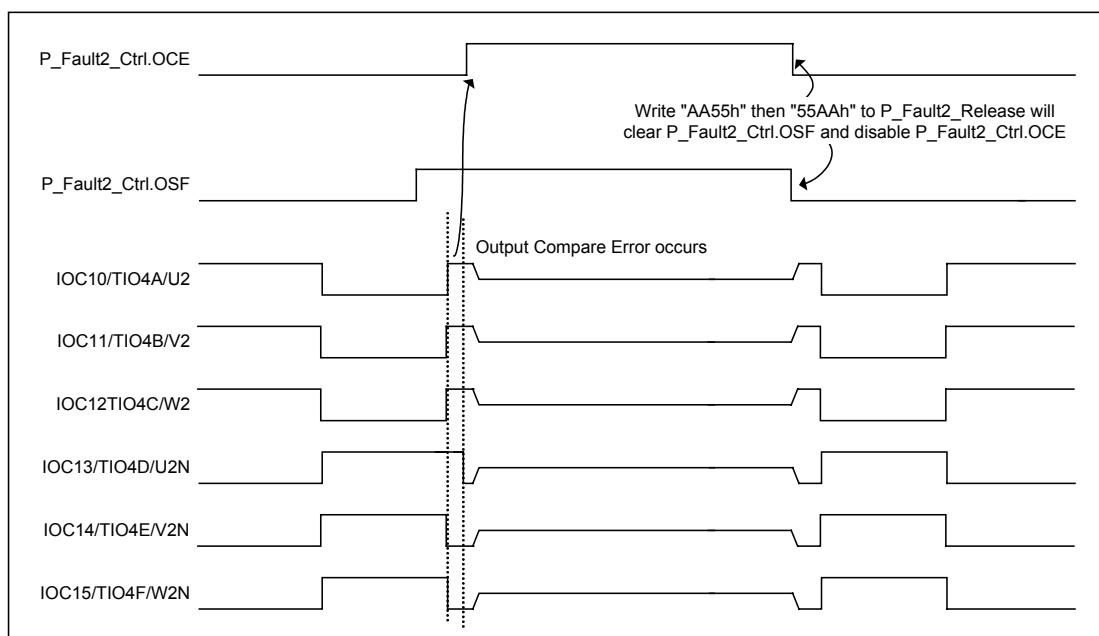


Figure 12-11 Output compare error timing

In MCP timer modules, the PWM outputs will work normally with correct clock source. Once the clock source is detected as fail, the PWM outputs will be immediately shut-down to high impedance state to prevent from damage the user's target board. The Figure 13-12 shows the oscillator stop timing.

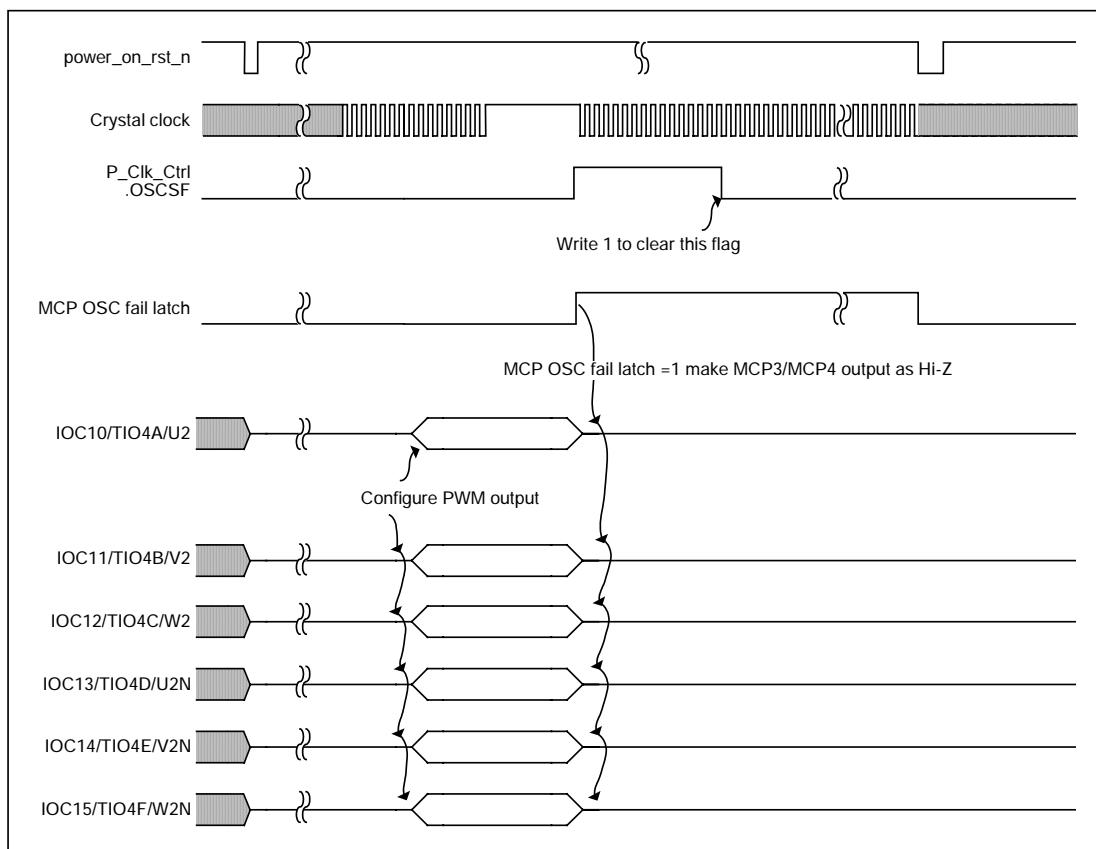


Figure 12-12 Oscillator stopped timing

OSF and FTPINIF can only be set to initial value by power-on reset but not by system reset. Fault protection state (PWM high-impedance state) can only be release by a power-on reset or software release procedures. External reset pin reset will not release this fault protection state!

Table 13-4 Fault input and PWM output pins combinations

Pin Name	Pin State	Description
FTIN1	Input	Input request to set U1, V1, W1, U1N, V1N, W1N output high-impedance
FTIN2	Input	Input request to set U2, V2, W2, U2N, V2N, W2N output high-impedance
PWM Output Pairs Combination	Pin State	Description
U1, U1N	Input Floating	All PWM output pins of MCP3 will be set to high-impedance state if these two pins output low-level simultaneously for more than one cycle
V1, V1N	Input Floating	
W1, W1N	Input Floating	
U2, U2N	Input Floating	All PWM output pins of MCP4 will be set to high-impedance state if these two pins output low-level simultaneously for more than one cycle
V2, V2N	Input Floating	
W2, W2N	Input Floating	

The PWM output will be halted (set to high-impedance state) under following circumstances.

Input 'low' to fault input pins (FTIN1/FTIN2)

The output level of upper phase output and lower phase output are the same

PLL or oscillator stopped

PWM output pins can be set to high-impedance state by FTIN1-2 pin falling edge or low-level sampling.

The valid FTINT input can be set for holding low for FCK/4 x 16, FCK/16 x 16, FCK/64 x 16, or FCK/256 x 16.

The complementary PWM output can be set to high-impedance state if upper and lower phase simultaneously output active-level for more than 1 system clock cycle.

Fast Interrupt Request (FIQ) will be generated if FTIP bit is set in P_INT_Priority register.

12.13.1 P_Fault1_Ctrl (0x7466): Fault input 1 Control and Status Register

12.13.2 P_Fault2_Ctrl (0x7467): Fault input 2 Control and Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R	R	R	R
0	0	0	0	0	0	0	0
OCE	OCIE	OCLS	OSF	Reserved			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FTPINE	FTPINIE	FTPINIF	Reserved	FTCNT			

Bit 15: **OCE:** Output compare enable. This bit enables/disables the output level comparison. If the output level compare function is enabled, the three pairs of PWM output will be compared. If any pair of the three pairs PWM output low-level simultaneously longer than one clock cycle (output short), all PWM outputs will go high-impedance. Write “AA55h” then “55AAh” to P_Faultx_Release (x = 1, 2) register will release the output compare protective state (PWM output in high-impedance state) and enable PWM output function. Also the OSF flag will be cleared. This bit will be read as ‘0’ when disabled.

0 = Disable

1 = Enable

Bit 14: **OCIE:** Output compare interrupt enable. This bit makes interrupt request if PWM output short has been detected.

0 = Disable

1 = Enable

Bit 13 **OCLS:** Output compare polarity level select. This bit selects the output compare level polarity.

0 = Compare low-level

1 = Compare high-level

Bit 12 **OSF:** Output short flag, This flag indicates that PWM output short has been detected. Write “AA55h” then “55AAh” to P_Faultx_Release (x = 1, 2) will clear this flag and also disable fault input pin. Software needs to set the OCE bit to ‘1’ again to active the output short protection function.

Bit 11:8 Reserved

Bit 7 **FTPINE:** Fault input pin 1/2 enable. This bit enables/disables FTINP1/2 pin input

0 = Disable

1 = Enable

Write “55AAh” then “AA55h” to FTRR1/2 will release the fault protective state (PWM output in high-impedance state) and disable FTINP1/2 pin input. Also FTPINIF flag will be cleared.

This bit will be read as ‘0’ when disabled.

- Bit 6 **FTPINIE:** Fault input 1/2 interrupt enable. This bit enables/disables FTINP1/2 pin interrupt.
 0 = Disable
 1 = Enable
- Bit 5 **FTPINIF:** Fault input 1/2 status flag. This bit indicates a high-impedance request has been input to FTINP1/2 pin. Write “55AAh” then “AA55h” to FTRR1/2 will clear the flag. Software needs to set the FTPINE bit to ‘1’ again to active the fault protection function.
 0 = Not occurred
 1 = Occurred
- Bit 4 Reserved
- Bit 3:0 **FTCNT:** Fault protection sampling time. FCK/4 * n, n = 1 to 15. User should note that setting the FTCNT value to 0 will always make the external fault input interrupt happen even the FTIN1/2 pin is at logic high state. If FTPINIE bit is set, FTPINIF will not be able to be cleared and the interrupt routine executed recursively due to incorrect FTCNT setting. This will cause the system unpredictable.

12.14 Timer Fault Release Register

To release the PWM output high-impedance state caused by fault input, first check the asserted fault pin input flag FTPINIF in P_Faultx_Ctrl (x = 1, 2) register, then write “55AAh” and “AA55h” sequentially to its corresponding fault release P_Faultx_Release (x = 1, 2) register.

To release the PWM output high-impedance state from PWM output short-circuit logic detection presents inside the chip, first check output short flag OSF in P_Faultx_Ctrl (x = 1, 2) register, then write “AA55h” and “55AAh” sequentially to its corresponding fault release P_Faultx_Release (x = 1, 2) register.

To release the PWM high-impedance state caused because oscillator fail, first clear oscillator fail flag OSCSF in P_Clk_Ctrl register, then write “5555h” and “AAAAh” sequentially to its corresponding fault release P_Faultx_Release (x = 1, 2) register.

12.14.1 P_Fault1_Release(0x746A): Fault 1 Flag Release Register

12.14.2 P_Fault2_Release(0x746B): Fault 2 Flag Release Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

Bit 15:0 **FTRR:** Fault release control words

12.15 Timer OverLoad Protection Control and Status Register

The SPMC75X family MCU devices contain an overload protection circuit. The circuit starts operating when the overload protection input (OL) is pulled low. The overload protection input is sampled by clock FCK/4. Sampling number can be set from 0 to 15 times.

There are three methods to deactivate overload protection: deactivate by a timer, deactivate by PWM synchronous, or deactivate manually. These methods can be used when the overload protection input has been released back to high.

The output disabled phases during overload protection are to disable no phases, all phases, PWM phases, or all upper/all lower phases. When to disable all upper or all lower phases is selected ($P_OLx_Ctrl.OLMD = 3, x = 1, 2$), motor drive PWM output is determined by their turn-on status immediately before being disabled. When two or more upper phases are active, all upper phases are turned on and all lower phases are turned off; when two or more lower phases are active, all upper phases are turned off and all lower phases are turned on. Table 13-5 and Table 13-6 show the overload function behavior depending on POLP bit in $P_TMRx_OutputCtrl (x = 3, 4)$ register and OLMD bit when such condition happened. To disable a phase means to put the phase in inactive level. Figure 13-13 shows the PWM output stopped timing when overload occurs.

Table 13-5 Overload protection interrupt when POLP = 1

POLP = 1		TIOxA ~ TIOxF Phase Output State (x = 3, 4)	Overload Protection Interrupt Capability
OLMD			
0	0	No phases disabled	No
0	1	All phases disabled	Yes
1	0	PWM/CPWM phases disabled. (Refer to Table 13-2)	Yes
1	1	(1) Any upper two phases are detected as high level, then disable all lower phases. (2) Any lower two phases are detected as high level, then disable all upper phases. If either condition (1) or (2) is not satisfied, phases are disabled.	Issue overload protection interrupt when any upper two phases or lower two phases are detected as high level, otherwise no interrupt is issued.

Table 13-6 Overload protection interrupt when POLP = 0

POLP = 1		TIOxA ~ TIOxF Phase Output State (x = 3, 4)	Overload Protection Interrupt Capability
OLMD			
0	0	No phases disabled	No
0	1	All phases disabled	Yes
1	0	PWM/CPWM phases disabled. (Refer to Table 13-3)	Yes
1	1	(1) Any upper two phases are detected as low level, then disable all lower phases. (2) Any lower two phases are detected as low level, then disable all upper phases. If either condition (1) or (2) is not satisfied, no phases are disabled.	Issue overload protection interrupt when Any upper two phase are detected as low level, otherwise no interrupt is issued.

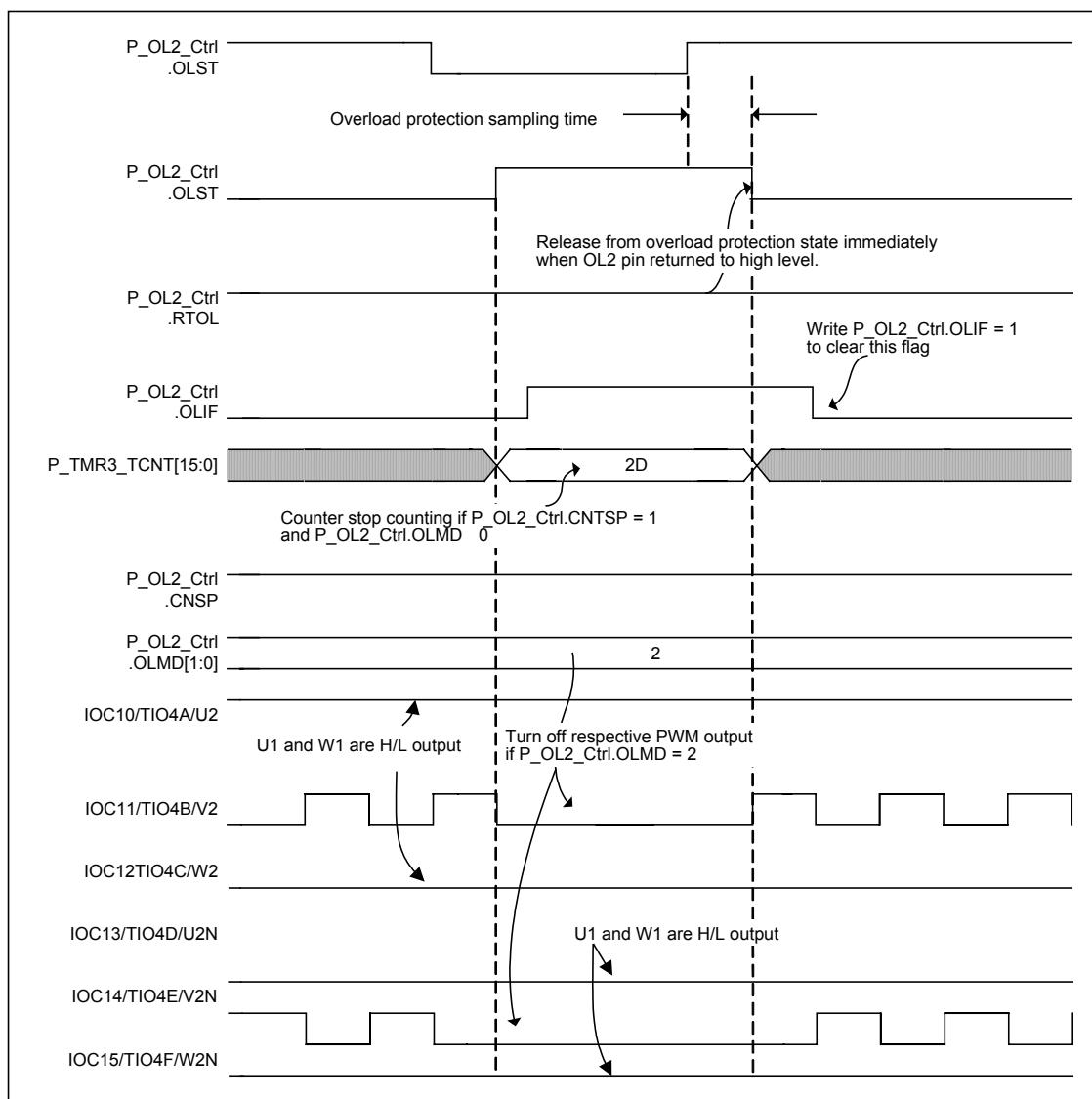


Figure 12-13 Stop PWM output only when overload occurs

12.15.1 P_Ol1_Ctrl(0x7468): Overload Input 1 Control and Status Register
12.15.2 P_Ol2_Ctrl(0x7469): Overload input 2 Control and Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLEN	CNTSP	OLMD		OLST	RTTMB	RTPWM	RTOL

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLIE	OLIF	Reserved			OLCNT		

Bit 15: **OLEN:** Overload protection enable. This bit enables/disables the overload protection circuit.

0 = Disable

1 = Enable

Bit 14: **CNTSP:** Stop PWM counter (P_TMR3_TCNT/P_TMR4_TCNT) during overload protection select.

0 = Do not stop

1 = Stop the counter

Bit 13:12 **OLMD:** Output disabled phases during overload protection select.

00 = No phases disabled

01 = All phases disabled, ie. on turn-off state

10 = PWM phases disabled

11 = All upper or all lower phases are disabled depending on the active phases

Bit 11 **OLST:** Overload protection status. This flag indicates the status of overload protection circuit.

0 = No operation

1 = Under protection

Bit 10 **RTTMB:** Deactivate overload protection from P_TMRx_TGRB (x = 0, 1) register compare match interrupt enable bit

0 = Disable

1 = Enable

Bit 9 **RTPWM:** Deactivate overload protection from delay one PWM period cycle when OLx (x = 1, 2) pin is brought to high level.

0 = Disable

1 = Enable

- Bit 8 **RTOL:** Deactivate overload protection from overload protection state
0 = No operation
1 = Automatically return from overload protection state if OLx (x = 1, 2) pin is brought to high level.
- Bit 7 **OLIE:** Overload interrupt enable bit
0 = Disable
1 = Enable
- Bit 6 **OLIF:** Overload interrupt flag. This flag indicates whether an overload condition has occurred. Write '1' to clear this flag.
0 = Not occurred
1 = Has occurred
- Bit 5:4 Reserved
- Bit 3:0 **OLCNT:** Overload protection sampling time. FCK/4 * n, n = 1 to 15. User should note that setting the OLCNT value to 0 will always make the external fault input interrupt happen even the FTIN1/2 pin is at logic high state. If OLIE bit is set, OLIF will not be able to be cleared and the interrupt routine executed recursively due to incorrect OLCNT setting. This will cause the system unpredictable.

12.16 Timer/PWM Module Write Enable Control Register

User must write 0x5A01 to P_TPWM_Write register to enable the timer 3 and write 0x5A02 to P_TPWM_Write to enable timer 4 for timer PWM generation. The P_TPWM_Write register provides a way to prevent the settings of timer 3 or 4 being miswritten due to CPU runaway. To modify the setting of timer 3 or timer 4, the corresponding TMR3/4WE bit must be set to '1'. Registers concerned with TMR3WE and TMR4WE are listed below. The recommended procedure of P_TPWM_Write register is first to read the content then do logical OR operation on control words (0x5A01 or 0x5A02). Write back the result to P_TPWM_Write last. The TMR3WE and TMR4WE control the MCP registers respectively as follows:

TMR3WE bit control the write operation of following registers :

P_TMR3_Ctrl, P_TMR3_IOCctrl, P_TMR3_INT, P_TMR3_Status, P_TMR3_DeadTime, P_TMR_Start,
P_TMR_Output

TMR4WE bit control the write operation of following registers :

P_TMR4_Ctrl, P_TMR4_IOCctrl, P_TMR4_INT, P_TMR4_Status, P_TMR4_DeadTime, P_TMR_Start,
P_TMR_Output

12.16.1 P_TPWM_Write (0x7409): Timer/PWM Module Write Enable Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
Reserved						TMR4WE	TMR3WE

Bit 15:2 Reserve

Bit 1 **TMR4WE**: Timer 4 setting registers write enable select bit

0 = Disable

1 = Enable

Bit 0 **TMR3WE**: Timer 3 setting registers write enable select bit

0 = Disable

1 = Enable

12.17 Timer Load-OK Register

In PWM output mode, to prevent partial duty parameters from being loaded incorrectly, correct updating procedures must be followed. The correct updating procedures are first update P_TMR3/4_TGRA-C, then set corresponding LDOK bit to '1'. Once LDOK bit has been set, all duty parameters are considered to be ready, and will be loaded to TGR when counter has been cleared. Then LDOK bit will be cleared to '0' when counter has been cleared. During LDOK be set to '1', the contents of P_TMR3/4_TGRA-C will not be altered by writing to these registers. To correctly set the LDOK bits, the pattern '101010' must be written to P_TMR_LDOK bit 7 to bit 2, otherwise the LDOK bits will not be updated. For example, to set LDOK0 to '1', 0x00A9 must be written to P_TMR_LDOK.

12.17.1 P_TMR_LDOK (0x740A) : Timer Load-OK Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TLDCHK						LDOK1	LDOK0

Bit15:8	Reserved
Bit 7:2	TLDCHK: Timer load register check bits To change the settings of P_TMR_LDOK, "101010" must be written to these bits. Otherwise LDOK1 and LDOK0 will not be changed. These bits will be read as '0'.
Bit 1	LDOK1: P_TMR4_TGRA-C ok to load bit This bit determines whether the values in P_TMR4_TGRA-C are ready to be loaded to PWM module. The values in P_TMR4_TGRA-C will not be loaded to PWM module until this bit has been set to '1'. After the values have been loaded, this bit will be cleared automatically. Note that when this bit has been set, the values in P_TMR4_TGRA-C will not be changed by writing to these registers.
Bit 0	LDOK0: P_TMR3_TGRA-C ok to load bit This bit determines whether the values in P_TMR3_TGRA-C are ready to be loaded to PWM module. The values in P_TMR3_TGRA-C will not be loaded to PWM module until this bit has been set to '1'. After the values have been loaded, this bit will be cleared automatically. Note that when this bit has been set, the values in P_TMR3_TGRA-C will not be changed by writing to these registers.

12.18 Timer Start Register

The P_TMR_Start register select the operation of counter start/stop for the P_TMRx_TCNT (x = 0 ~ 4). When counter operation stopped, its contents will be cleared. Set TMR3ST or TMR4ST bit to 1 would start the P_TMR3_TCNT or P_TMR4_TCNT register immediately and vice versa.

12.18.1 P_TMR_Start (0x7405): Timer Counter Start Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved			TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST

Bit 15:5 Reserved

Bit 4 **TMR4ST:** Timer 4 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

Bit 3 **TMR3ST:** Timer 3 counter start setting

0 = Counter operation stopped

1 = Performs counting operation

- Bit 2 **TMR2ST:** Timer 2 counter start setting
 0 = Counter operation stopped
 1 = Performs counting operation
- Bit 1 **TMR1ST:** Timer 1 counter start setting
 0 = Counter operation stopped
 1 = Performs counting operation
- Bit 0 **TMR0ST:** Timer 0 counter start setting
 0 = Counter operation stopped
 1 = Performs counting operation

12.19 Timer 3 and 4 Counter Register

The MCP timer 3 and timer 4 have two TCNT counters (P_TMR3_TCNT and P_TMR4_TCNT), one for each channel. The TCNT counters are 16-bit readable registers that increment/decrement according to input clocks.

Bits TMRPS in corresponding timer control register can select input clocks. P_TMR3_TCNT and P_TMR4_TCNT increment/decrement in center-aligned PWM mode, while they only increment in other modes. The TCNT counters are initialized to 0x0000 when TCNT value matches the period register.

12.19.1 P_TMR3_TCNT (0x7433): Timer 3 Counter Register

12.19.2 P_TMR4_TCNT (0x7434): Timer 4 Counter Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

12.20 Timer 3 and 4 General and Buffer Register

TGRA, TGRB, TGRC, TGRD are 16-bit registers. MCP timer 3 and 4 has totally eight timer general registers, four for each channel. The TGRA, TGRB and TGRC registers are dual function 16-bit readable/writable registers, functioning as output compare or PWM registers. The TGRD register uses as the ADC conversion start signal when TCNT counter value match this register content. The values in TGR and TCNT are constantly compared with each other when the TGR registers are used as output compare or PWM registers.

When PWM mode, edge-aligned PWM mode, or center-aligned PWM mode is selected, the TGR register behaves as the duty ratio value register. Upon reset, the TGR registers are initialized to 0x0000.

The timer buffer registers TBRA, TBRB and TBRC are the double buffers of TGRA, TGRB and TGRC, respectively. The value of TGR_x (x=A, B, C) can automatically be updated when the period compare match event occurs. That is, the duty ratio value will not be updated until one period ends completely.

12.20.1 P_TMR3_TGRA (0x7448): Timer 3 General Register A

12.20.2 P_TMR3_TGRB (0x7449): Timer 3 General Register B

12.20.3 P_TMR3_TGRC (0x744A): Timer 3 General Register C

12.20.4 P_TMR3_TGRD (0x744B): Timer 3 General Register D

12.20.5 P_TMR4_TGRA (0x744C): Timer 4 General Register A

12.20.6 P_TMR4_TGRB (0x744D): Timer 4 General Register B

12.20.7 P_TMR4_TGRC (0x744E): Timer 4 General Register C

12.20.8 P_TMR4_TGRD (0x744F): Timer4 General Register D

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

12.20.9 P_TMR3_TBRA (0x7458): Timer 3 Buffer Register A

12.20.10 P_TMR3_TBRB (0x7459): Timer 3 Buffer Register B

12.20.11 P_TMR3_TBRC (0x745A): Timer 3 Buffer Register C

12.20.12 P_TMR4_TBRA (0x745C): Timer 4 Buffer Register A

12.20.13 P_TMR4_TBRB (0x745D): Timer 4 Buffer Register B

12.20.14 P_TMR4_TBRC (0x745E): Timer 4 Buffer Register C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

12.21 Timer 3 and 4 Period Register

The P_TMRx_TPR ($x = 3, 4$) is a 16-bit readable/writable register. It is used to set the period of PWM waveform. When P_TMRx_TCNT ($x = 3, 4$) register reaches P_TMRx_TPR ($x = 3, 4$) register value, P_TMRx_TCNT ($x = 3, 4$) register will be cleared to 0x0000 (up-counting mode) or start down-count (continuous up-/down-counting mode) according to MODE bits programmed in P_TMRx_Ctrl ($x = 3, 4$) registers. Its default value is 0xFFFF. When P_TMRx_TPR ($x = 3, 4$) register is set to 0x0000, the P_TMRx_TCNT ($x = 3, 4$) register counter will stop counting and remain at 0x0000.

12.21.1 P_TMR3_TPR (0x7438): Timer 3 Period Register

12.21.2 P_TMR4_TPR (0x7439): Timer 4 Period Register

TMRPRD

TMRPRD

12.22 MCP Timer 3 and 4 Operation

12.22.1 Normal Counting Operation

When TMR3ST or TMR4ST bit is set in P_TMR_Start register, the P_TMRx_TCNT ($x = 3, 4$) register for the corresponding channel beginning up direction counting. The counter register behaves as a free running operation and would be reset to 0x0000 when reaches 0xFFFF. Through configures the CCLS bits to 111'b, the timers is setup as periodic counter. The counter register would be reset to 0x0000 when reaches the value of P_TMRx_TPR ($x = 3, 4$) registers. Figure 13-14 shows the programming flowchart of normal counting operation.

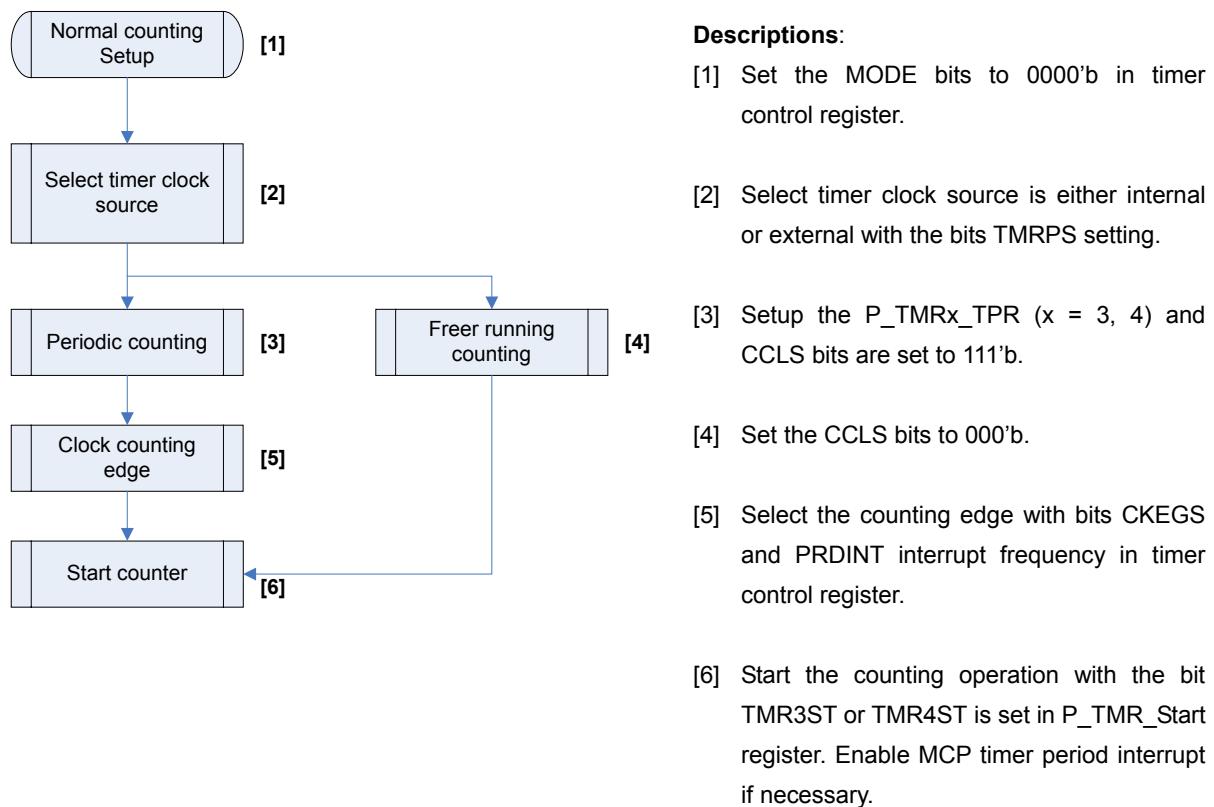


Figure 12-14 Example programming flowchart of normal counting operation

The initial value of P_TMRx_TCNT ($x = 3, 4$) is 0x0000 and P_TMRx_TPR ($x = 3, 4$) is 0xFFFF. When the corresponding bit TMR3ST or TMR4ST is set to 1, the P_TMRx_TCNT ($x = 3, 4$) behaves as the free-running counter.

When a period compare match event is selected as P_TMRx_TCNT ($x = 3, 4$) register clearing source, the counter behaves as the periodic counting operation. The P_TMRx_TPR ($x = 3, 4$) registers for setting the period the period and counter-clearing source is selected by setting CCLS bits to 111'b. After the settings have made, the counter register start an increment operation as periodic counter when corresponding bit TMR3ST or TMR4ST is set to 1. When counter matches the value of P_TMRx_TPR ($x = 3, 4$) register, the TPRIIF flag is set to 1 in timer interrupt status register and counter register is reset to 0x0000. If the bit value of TPRIE in P_TMRx_INT ($x = 3, 4$) register is set to 1, the MCP timer requests an interrupt.

12.22.2 PWM Output Operation

The MCP timer module has two channels and can perform PWM function up to twelve pins output. The output waveforms have active low at compare match, active high at compare match , forced high and forced low for the corresponding TIOxA, TIOxB, TIOxC, TIOxD, TIOxE and TIOxF ($x = 3, 4$) output pin using compare match with P_TMRx_TGRA, P_TMRx_TGRB, P_TMRx_TGRC ($x = 3, 4$) register respectively. Figure 13-15 shows the programming flowchart of PWM operation.

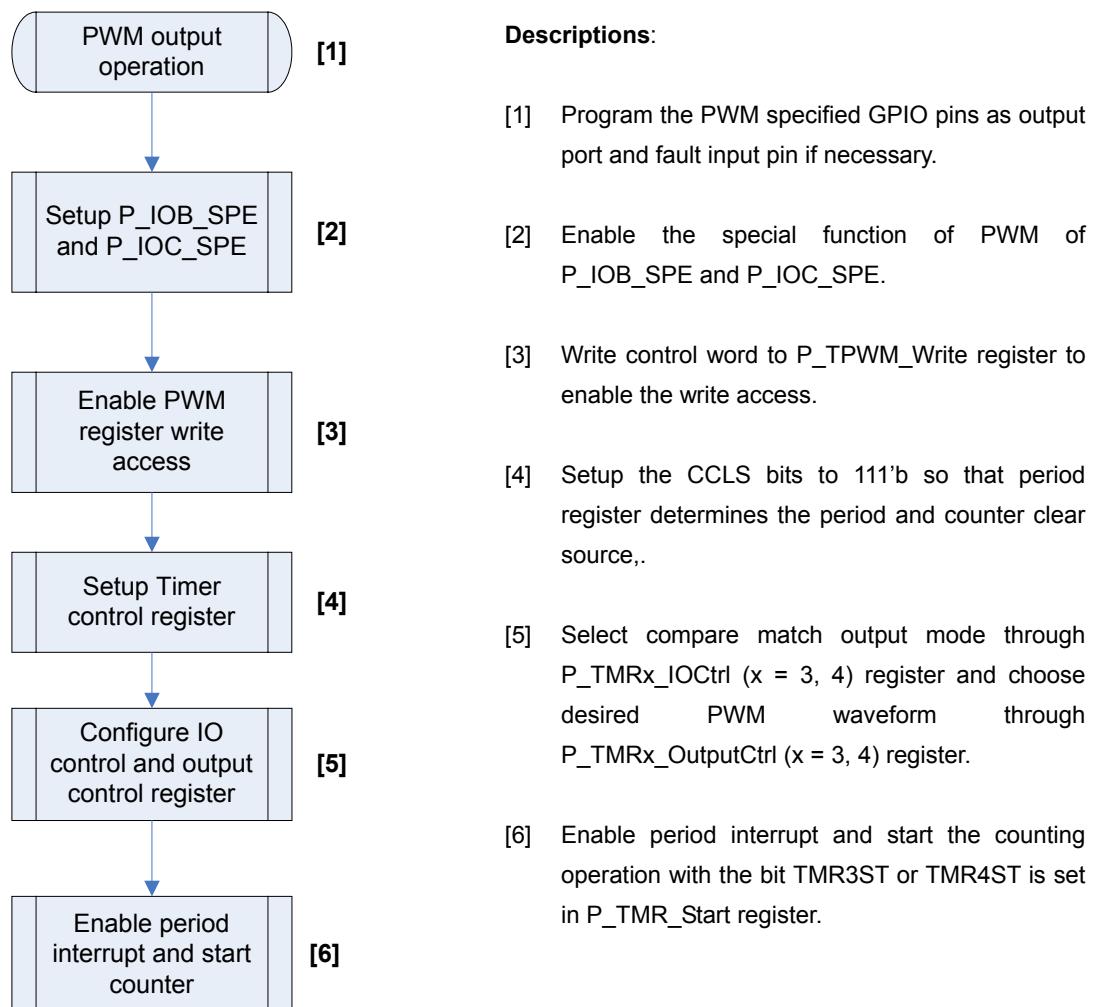


Figure 12-15 Example programming flowchart of PWM operation

12.23 Application Example Design Tips

【Example 13-1】: How to set the timer 3 frequency?

The timer 3 frequency is depended on the timer 3 source clock, counter clock edge and period interrupt frequency with the bits setting TMRPS, CKEGS and PRDINT. The clock source fed into timer 3 has eight sources. They are Fck/1, Fck/4, Fck/16, Fck/64, Fck/256, Fck/1024, TCLKA and TCLKB. CKEGS bits select the input clock edge. When the input clock is counted using both edges, the input clock period is halved. When FCK/1 is selected as counter clock, counter will count at rising edge if count at both edges is selected. Also the PRDINT setting will affected the timer 3 period interrupt frequency, After these setting are done, the P_TMR3_PRD data should be written to.

For instances, if we intended to set the timer 3 period interrupt frequency as 8K Hz :

$$P_TMR3_TPR = (\text{source clock} / 8000) - 1$$

```

/* Timer 3 interrupt frequency is 8000 Hz ,           */
/* select timer 3 clock source as Fck/1 = 20MHz. */

#include "Spmc75_regs.h"

int main(void)
{
    Disable_FIQ_IRQ();

    P_TPWM_Write->W = P_TPWM_Write->W | 0x5A01;      /* enable TMR3 module */
    P_TMR3_Ctrl->B.CCLS = CB_TMR3_CCLS_PTR;          /* clear on TPR match */
    P_TMR3_Ctrl->B.TMRPS = CB_TMR3_TMRPS_FCKdiv1;   /* count on FCK/1, rising edge */
    P_TMR3_Ctrl->B.CKEGS = CB_TMR0_CKEGS_Rising;     /* count on rising */
    P_TMR3_Ctrl->B.MODE = CB_TMR3_MODE_Normal;       /* normal counter mode */
    P_TMR3_Ctrl->B.PRDINT = CB_TMR3_PRDINT_Period;  /* interrupt every period */
    P_TMR3_TCNT->W = 0x0000;                          /* reset TMR3 counter */
    P_TMR3_TPR->W = 2499;                            /* 2499 setting for 8000 Hz */

    P_TMR3_INT->B.TPRIE = 1;                         /* enable TPR3 interrupt */
    P_TMR_Start->W = CW_TMR_TMR3ST_Start;            /* start TMR3 */
    INT_FIQ_IRQ();                                     /* enable FIQ/IRQ channel interrupt */

    while(1);
    return (1);
}

/*****************************************/
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
    if(P_TMR3_Status->B.MCP3IF)
    {
        P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;      /* clear TMR3 TPRIF */
                                                       /* do TMR3_TPR ISR tasks */
    }
}

```

【Example 13-2】: How to generate the 120-degree six-step waveform for driving BLDC frequency?

First we need set the TPR interrupt frequency of MCP timer 3. The 120-degree six-step waveform is determined by fetch in motor position hall signal of PDC timer module, please refer to section 11. Timer 3 interrupt frequency is 8000 Hz , select timer 3 clock source as FCK/1 = 24MHz. The three-phase duty cycle register use only the TGRA register. No dead-time feature is inserted.

```

#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{

```

```
Disable_FIQ_IRQ();
Init_TMR3 ();                                /* initialize TMR3 module */

P_IOB_SPE->W = 0x003F;                      /* enable UVW-1 output special function */
P_TMR3_IOCctrl->W = 0x0111;                  /* TIO3A ~ TIO3F output enable */
P_TMR3_OutputCtrl->B.DUTYMODE = 0;
                                                /* three phase in common, use TGRA register */

P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;      /* active high */
P_POS0_DectCtrl->W = 0x0000;                  /* sampling clock = FCK/4 */
P_POS0_DectCtrl->B.SPLMOD = 1;                /* sample regularly */
P_POS0_DectCtrl->B.PDEN = 1;                  /* enable position detection function */

P_TMR_Start->W = CW_TMR_TMR3ST_Start;        /* start TMR3 */
INT_FIQ_IRQ();                                /* enable FIQ/IRQ channel interrupt */

while(1);
return (1);
}

/*****************************************/
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
    int position;
    if(P_TMR3_Status->B.TPRIF)
    {
        P_TMR3_Status->W = CW_TMR3_TPRIF_Enable; /* clear TMR3 TPRIF */

        /* generate 120-degree six-step PWM waveform according to detection */
        /* position signal, the following switch-case demo codes may be different with
           developer's application */
        position = P_POS0_DectData->W;
        switch(position)
        {
            case (1):
                P_TMR3_OutputCtrl->B.UPWM = 0;
                P_TMR3_OutputCtrl->B.UOC = 0;      /* U1 = L, U1N = L */

                P_TMR3_OutputCtrl->B.VPWM = 0;
                P_TMR3_OutputCtrl->B.VOC = 1;      /* V1 = L, V1N = H */

                P_TMR3_OutputCtrl->B.WPWM = 1;
                P_TMR3_OutputCtrl->B.WOC = 2;      /* W1 = PWM, W1N = L */
                break;
        }
    }
}
```

```
case (2):
    P_TMR3_OutputCtrl->B.UPWM = 1;
    P_TMR3_OutputCtrl->B.UOC = 2;      /* U1 = PWM, U1N = L */

    P_TMR3_OutputCtrl->B.VPWM = 0;
    P_TMR3_OutputCtrl->B.VOC = 1;      /* V1 = L, V1N = H */

    P_TMR3_OutputCtrl->B.WPWM = 0;
    P_TMR3_OutputCtrl->B.WOC = 0;      /* W1 = L, W1N = L */
    break;

case (3):
    P_TMR3_OutputCtrl->B.UPWM = 1;
    P_TMR3_OutputCtrl->B.UOC = 2;      /* U1 = PWM, U1N = L */

    P_TMR3_OutputCtrl->B.VPWM = 0;
    P_TMR3_OutputCtrl->B.VOC = 0;      /* V1 = L, V1N = L */

    P_TMR3_OutputCtrl->B.WPWM = 0;
    P_TMR3_OutputCtrl->B.WOC = 1;      /* W1 = L, W1N = H */
    break;

case (4):
    P_TMR3_OutputCtrl->B.UPWM = 0;
    P_TMR3_OutputCtrl->B.UOC = 0;      /* U1 = L, U1N = L */

    P_TMR3_OutputCtrl->B.VPWM = 1;
    P_TMR3_OutputCtrl->B.VOC = 2;      /* V1 = PWM, V1N = L */

    P_TMR3_OutputCtrl->B.WPWM = 0;
    P_TMR3_OutputCtrl->B.WOC = 1;      /* W1 = L, W1N = H */
    break;

case (5):
    P_TMR3_OutputCtrl->B.UPWM = 0;
    P_TMR3_OutputCtrl->B.UOC = 1;      /* U1 = L, U1N = H */

    P_TMR3_OutputCtrl->B.VPWM = 1;
    P_TMR3_OutputCtrl->B.VOC = 2;      /* V1 = PWM, V1N = L */

    P_TMR3_OutputCtrl->B.WPWM = 0;
    P_TMR3_OutputCtrl->B.WOC = 0;      /* W1 = L, W1N = L */
    break;

case (6):
```

```

P_TMR3_OutputCtrl->B.UPWM = 0;
P_TMR3_OutputCtrl->B.UOC = 1;           /* U1 = L, U1N = H */

P_TMR3_OutputCtrl->B.VPWM = 0;
P_TMR3_OutputCtrl->B.VOC = 0;           /* V1 = L, V1N = L */

P_TMR3_OutputCtrl->B.WPWM = 1;
P_TMR3_OutputCtrl->B.WOC = 2;           /* W1 = PWM, W1N = L */
break;
}

P_TMR3_TGRA->W = 750;                  /* set duty ratio = 30 % for three phase */
P_TMR_LDOK->W = P_TMR_LDOK->W | CW_TMR_LDOK0;
                                         /* load to TGRA ~ TGRC simultaneously */
}
}

```

【Example 13-3】: How to setup the dead-time inserted on the U1/U1N, V1/V1N and W1/W1N transistor pairs?

Dead-time generation feature is to protect the three phase power inverter not be damaged by short-circuit fault. To active the dead-time inserted, the P_TMRx_DeadTime (x = 3, 4) register must be properly programmed. And also program the P_TMRx_OutputCtrl (x = 3, 4) UOC/VOC/WOC bits to select complementary PWM mode. The following is the sample code listing for three-phase power inverter application.

```

#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{
    Disable_FIQ_IRQ();
    Init_TMR3();                         /* initialize TMR3 module */

    P_IOB_SPE->W = 0x003F;               /* enable UVW-1 output special function */
    P_TMR3_IOCctrl->W = 0x0111;          /* TIO3A ~ TIO3F output enable */
    P_TMR3_OutputCtrl->B.DUTYMODE = 0;
                                         /* three phase in common, use TGRA register */
    P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High;      /* active high */
    P_TMR_Start->W = CW_TMR_TMR3ST_Start;                      /* start TMR3 */
    INT_FIQ_IRQ();                         /* enable FIQ/IRQ channel interrupt */

    while(1);
    return (1);
}
/*********************************************

```

```

void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
    if(P_INT_Status->B.MCP3IF)
    {
        P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;           /* clear TMR3 TPRIF */
        P_TMR3_DeadTime->B.DTWE = 1;                      /* enable dead-time timer for W phase */
        P_TMR3_DeadTime->B.DTVE = 1;                      /* enable dead-time timer for V phase */
        P_TMR3_DeadTime->B.DTUE = 1;                      /* enable dead-time timer for U phase */
        P_TMR3_DeadTime->B.DTP = 12;                     /* 2.0us at Fck / 4 clock, FCK = 20.0 MHz */

        /* sample code to inserted dead-time on W-phase PWM */
        P_TMR3_OutputCtrl->W = (P_TMR3_OutputCtrl->W & 0xF800) | \
                                (CW_TMR3_POLP_1_UP_L_UN_L | \
                                 CW_TMR3_POLP_1_VP_L_VN_H | \
                                 CW_TMR3_POLP_1_WP_PWM_WN_CPWM);
    }
}

```

【Example 13-4】: Setup the fault input pin 1 function to protect MCP3 six phase PWM. The following code lists the sample code.

```

#include "Spmc75_regs.h"
#include "unspmacro.h"

int main(void)
{
    Disable_FIQ_IRQ();
    Init_TMR3();                         /* initialize TMR3 module */

    P_IOB_SPE->W = 0x003F;              /* enable UVW-1 output special function */
    P_TMR3_IOCctrl->W = 0x0111;         /* TIO3A ~ TIO3F output enable */
    P_TMR3_OutputCtrl->B.DUTYMODE = 0;
                                         /* three phase in common, use TGRA register */
    P_TMR3_OutputCtrl->B.POLP = CB_TMR3_POLP_Active_High; /* active high */

    P_IOB_SPE |= CW_IOB_FTIN1_SFR_EN; /* enable FTIN1 special function */
    P_Fault1_Ctrl->B.OCE = 1;          /* enable output level comparsion */
    P_Fault1_Ctrl->B.OCIE = 1;          /* enable output compare interrupt */

    P_Fault1_Ctrl->B.OCLS = 1;          /* compare high level */
    P_Fault1_Ctrl->B.FTPINE = 1;        /* enable FTIN1 pin */
    P_Fault1_Ctrl->B.FTPINIE = 1;       /* Fault 1 interrupt enable */
    P_Fault1_Ctrl->B.FTCNT = 1;
    P_INT_Priority->B.FTIP = 1;         /* FTIN1 assign to FIQ */

```

```

        while(1);
        return (1);
    }

/*****void FIQ(void) __attribute__ ((ISR));
void FIQ(void)
{
    if(P_INT_Status->B.FTIF)
    {
        if(P_Fault1_Ctrl->B.OSF)
        {
            P_Fault1_Release->W = 0xAA55;           /* clear OSF flag */
            P_Fault1_Release->W = 0x55AA;
            P_Fault1_Ctrl->B.FTPINE = 1;           /* re-enable FTIN1 pin */
        }

        if(P_Fault1_Ctrl->B.FTPINIF)
        {
            P_Fault1_Release->W = 0x55AA;          /* clear FTPINIF flag */
            P_Fault1_Release->W = 0xAA55;
            P_Fault1_Ctrl->B.FTPINE = 1;           /* re-enable FTIN1 pin */
        }
    }
}

```

【Example 13-5】: Setup the over-loading protection function for MCP3 unit driving PWM.

```

void init_OLP3(void)
{
    P_OL1_Ctrl->B.OLEN = 1;           /* enable overload protection */
    P_OL1_Ctrl->B.OLMD = 2;           /* disable PWM phase when OL occured */
    P_OL1_Ctrl->B.OLCNT = 1;          /* OL sampling time counter= 1 at Fck/4 */
    P_OL1_Ctrl->B.RTOL = 1;
}

/*****void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
    if(P_INT_Status->B.MCP3IF)
    {
        P_TMR3_Status->W = CW_TMR3_TPRIF_Enable;      /* clear TMR3 TPRIF */

        if(P_OL1_Ctrl->B.OLST)                         /* over-load protection occurred ? */
        {

```

```
// record the over-loading event history in your software.  
// Then execute the necessary tasks.  
}  
}  
}
```

【Example 13-6】: Setup the over-loading protection deactivate from PWM synchronous enable bit.

```
void init_OLP3(void)  
{  
    P_OL1_Ctrl->W = (CW_TMR3_OPR_OLEN | CW_TMR3_OPR_OLMDis | \  
                        CW_TMR3_OPR_RTPWM_Enable | CW_TMR3_OPR_OLEN);  
    P_OL1_Ctrl->B.OLCNT = 1;  
        /* overload protection enable */  
        /* all phase disable during OL */  
        /* deactivate OL from PWM enable */  
        /* OL sampling time counter = 1 at FCK / 4 */  
}
```

【Example 13-7】 : AC induction motor with sinusoidal waveform MCP3 register setup procedure.

The SPMC75X family MCU chip supports both edge and center aligned PWM waveform generation. In this example, the center aligned PWM mode is suitable for AC induction motor driving with sinusoidal wave due to fewer harmonics.

The PWM pulse width of three phase is determined by P_TMR3_TGRA, P_TMR3_TGRB and P_TMR3_TGRC. During each sinusoidal period, these register value are fetched from lookup table stored inside the program memory of CPU. If the timer general register TGR matches the timer counter P_TMR3_TCNT, the corresponding port is triggered an output event according to P_TMR3_OutputCtrl register. The dead time interval between upper and lower phase is determined through P_TMR3_DeadTime register, please refer to figure 13-6 for the reference waveform.

For three phase AC inverter with sinusoidal waveform, the triangular wave frequency is the PWM carrier frequency which is the determined from CPU clock frequency FCK and period register P_TMR3_TPR. The sequence of different pulse width of PWM can form one sinusoidal period. The following flowchart shows the setting procedure for the AC induction motor with sinusoidal waveform application.

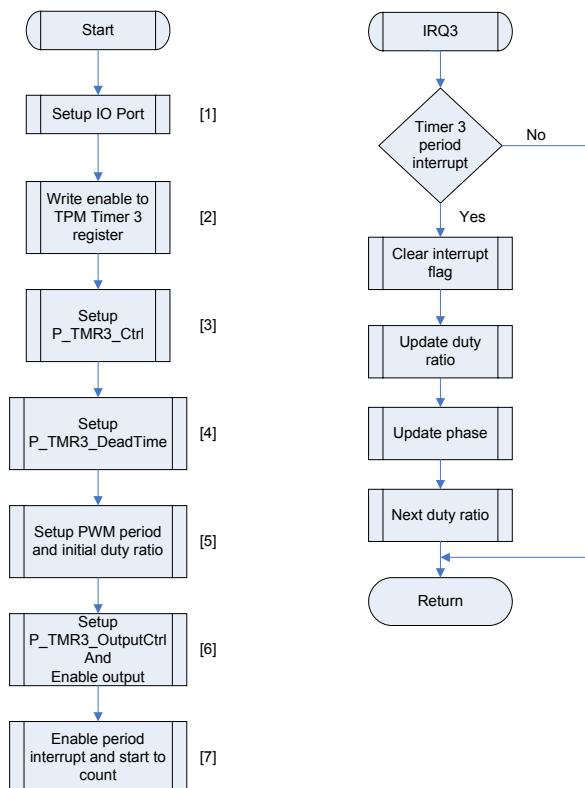


Figure 12-16 Example programming flowchart of sinusoidal PWM operation

Descriptions:

[1] Setup P_IOB_Dir for PWM output port and enable special function of P_IOB_SPE.

[2] Enable to write MCP timer 3 registers :

P_TPWM_Write->W = P_TPWM_Write->W | 0x5A01;

[3] Setup MCP timer 3 control register, P_TMR3_Ctrl :

PRDINT = 0, interrupt every period and timer period is determined from P_TMR3_TPR and CPU operating clock frequency,

MODE = 1010'b, set to center aligned PWM mode.

CCLS = 111'b, setup clear source of MCP timer 3.

CKEGRS = 0, count on rising edge.

TMRPS = 0, timer clock source is FCK

[4] Setup P_TMR3_DeadTime to 0x7001, enable three phase dead time.

[5] Setup PWM period and initial duty ratio.

P_TMR3_TPR->W = 0x0960, determines the PWM period.

P_TMR3_TGRA->W = 0x0400, phase U duty ratio

P_TMR3_TGRB->W = 0x0766, phase V duty ratio

P_TMR3_TGRC->W = 0x0088, phase W duty ratio

[6] Setup MCP timer 3 PWM output mode and enable corresponding output pin.

P_TMR3_OutputCtrl->W = 0x8700, three phase independent PWM output.

P_TMR3_IOCtrl->W = 0x0111, initial 0 and 1 at compare match.

P_TMR_Output->W = 0x003F, enable MCP timer 3 PWM output.

[7] Enable MCP timer 3 period interrupt and start to count.

P_TMR3_INT->B.TPRIE = 1, enable period interrupt.

P_TMR_Start->B.TMR3ST = 1, timer 3 start to count.

The following code shows the procedure for the whole operation described above.

```
#include "Spmc75_regs.h"
#include "Spmc_typedef.h"
#include "unspMACRO.h"
extern int Sin_TAB[];
UInt16 SPWM_phases;                                /* Phase increment */
UInt16 Phases_Temp;                                /* lookup table index */
UInt16 Phases_Add_Data;                            /* phase delta */
int NEW_Data[3];
void Sys_Init(void);
void CCP_Init(void);

int main(void)
{
    Sys_Init();
    CCP_Init();
    INT_IRQ();
    while(1);
}

/*****************************************/
void Sys_Init(void)
{
    while(P_Clk_Ctrl->B.OSCSF & 1)          /* check Oscillator Sralus Flag */
    {
        P_Clk_Ctrl->B.OSCSF = 1;            /* Clear Oscillator Status Flag */
    }
    P_System_Option->W = 0x5551           /* with external oscillator */
    SPWM_phases = 0;
    Phases_Add_Data = 656;                 /* reference frequency = 50Hz */
}
```

```

/*****
void CCP_Init(void)
{
    P_IOB_Dir->W = 0x0000;                                /* IO seting */
    P_IOB_Attrib->W = 0xffff;
    P_IOB_Data->W = 0xff00;
    P_IOB_SPE->W = 0x00ff;
    P_TPWM_Write->W = P_TPWM_Write->W | 0x5A01; /* Enable Ctrl register Write. */
    P_TMR3_OutputCtrl->W = 0x8700; /* Three phases independent output Ctrl */
    P_TMR3_IOCtrl->W = 0x0111;
    P_TMR3_DeadTime->W = 0x7001;           /* select dead-time */
    P_TMR3_Ctrl->B.PRDINT = 0;
    P_TMR3_Ctrl->B.MODE = 10;
    P_TMR3_Ctrl->B.CCLS = 7;
    P_TMR3_Ctrl->B.CKEGS = 0;
    P_TMR3_Ctrl->B.TMRPS = 0;
    P_TMR_Output->W = 0x3f3f;             /* Enable Output pin */
    P_TMR3_TPR->W = 0x0960;               /* Timer Period setting */
    P_TMR3_TGRA->W = 0x0400;
    P_TMR3_TGRB->W = 0x0766;
    P_TMR3_TGRC->W = 0x0088;
    P_TMR3_INT->W = 0x0010;              /* Timer Period interrupt Enable */
    P_TMR_Start->B.TMR3ST = 1;           /* Timer 3 RUN */
}

/*****
void IRQ3(void) __attribute__ ((ISR));
void IRQ3(void)
{
    if(P_TMR3_Status->B.TPRIF)
    {
        P_TMR3_Status->B.TPRIF = 1;          /* Clear TPRIF flag */
        P_TMR3_TGRA->W = NEW_Data[0];       /* update data */
        P_TMR3_TGRB->W = NEW_Data[1];
        P_TMR3_TGRC->W = NEW_Data[2];
        P_TMR_LDOK->W = P_TMR_LDOK->W | CW_TMR_LDOK0;
        SPWM_phases += Phases_Add_Data;      /* add phase */
        Phases_Temp = SPWM_phases>>6
        NEW_Data[0] = Sin_TAB[Phases_Temp];
        NEW_Data[1] = Sin_TAB[(Phases_Temp+341)&0x03ff];
        NEW_Data[2] = Sin_TAB[(Phases_Temp+683)&0x03ff];
    }
}

```

13 Compare Match Timer

13.1 Introduction

SPMC75X family MCU has a compare match timer (CMT) comprising two 16-bit timer channels. Each channel has a 16-bit up-count counter and can generate interrupt at set intervals. The clock input source can be selected from $F_{CK}/1$, $F_{CK}/2$, $F_{CK}/4$, $F_{CK}/8$, $F_{CK}/16$, $F_{CK}/64$, $F_{CK}/256$, or $F_{CK}/1024$. The compare match interrupt will be set if the register value of P_{CMTx_TCNT} ($x=0, 1$) matches that of P_{CMTx_TPR} ($x = 0, 1$), respectively. The counters will start counting when STx ($x = 0, 1$) in P_{CMT_Start} is set, independently.

Features

- 8 counter input clocks can be selected
 - $F_{CK}/1$, $F_{CK}/2$, $F_{CK}/4$, $F_{CK}/8$, $F_{CK}/16$, $F_{CK}/64$, $F_{CK}/256$, $F_{CK}/1024$
- Interrupt sources
 - Each channel has its independent compare match interrupt

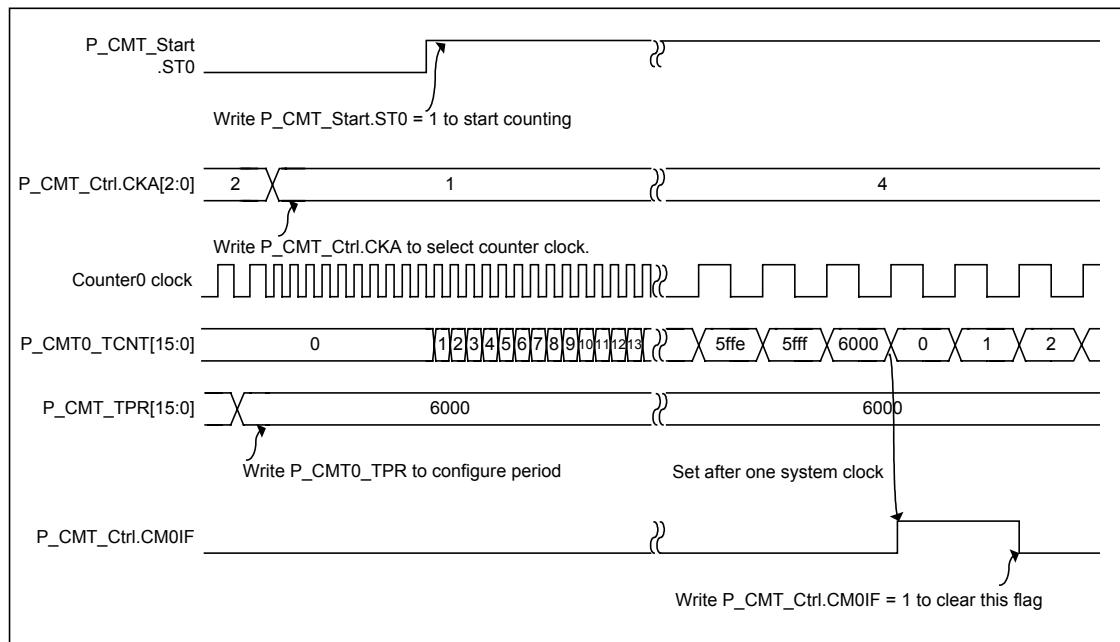


Figure 13-1 CMT timing

13.2 Compare Match Timer Registers Address Table

Table 14-1 Compare match timer registers

Address	Register	Name
7500h	P_CMT_Start	Compare match timer start register
7501h	P_CMT_Ctrl	Compare match timer control and status register
7508h	P_CMT0_TCNT	Compare match timer 0 counter register
7509h	P_CMT1_TCNT	Compare match timer 1 counter register
7510h	P_CMT0_TPR	Compare match timer 0 period register
7511h	P_CMT1_TPR	Comapre match timer 1 period register

13.3 Control Registers

13.3.1 P_CMT_Start (0x7500) : Compare Match Timer Start Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
Reserved						ST1	ST0

Bit 15:2: Reserved

Bit 1: **ST1**: Compare match timer 1 counter start

0 = P_CMT1_TCNT counter operation stopped, and cleared to 0x0000

1 = P_CMT1_TCNT counter operation enabled

Bit 0: **ST0**: Compare match timer 0 counter start

0 = P_CMT0_TCNT counter operation stopped, and cleared to 0x0000

1 = P_CMT0_TCNT counter operation enabled

13.3.2 P_CMT_Ctrl (0x7501) : Compare Match Timer Control and Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM1IF	CM1IE	Reserved			CKB		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM0IF	CM0IE	Reserved			CKA		

- Bit 15: **CM1IF**, CMT1 compare match interrupt flag, this flags indicates whether the P_CMT1_TCNT and P_CMT1_TPR value have matched. Write '1' to clear this flag.
- 0 = Not matched
1 = Matched
- Bit 14: **CM1IE**, CMT1 compare match interrupt enable
- 0 = CMT1 compare match interrupt disable
1 = CMT1 compare match interrupt enable
- Bit13:11: Reserved
- Bit 10:8: **CKB**: CMT1 clock select bits
- | | |
|-----|--------------|
| 000 | = FCK / 1 |
| 001 | = FCK / 2 |
| 010 | = FCK / 4 |
| 011 | = FCK / 8 |
| 100 | = FCK / 16 |
| 101 | = FCK / 64 |
| 110 | = FCK / 256 |
| 111 | = FCK / 1024 |
- Bit 7: **CM0IF**, CMT0 compare match interrupt flag, this flags indicates whether the P_CMT0_TCNT and P_CMT0_TPR value have matched. Write '1' to clear this flag.
- 0 = Not matched
1 = Matched
- Bit 6: **CM0IE**, CMT0 compare match interrupt enable
- 0 = CMT0 compare match interrupt disable
1 = CMT0 compare match interrupt enable
- Bit5:3: Reserved
- Bit 10:8: **CKA**: CMT0 clock select bits
- | | |
|-----|--------------|
| 000 | = FCK / 1 |
| 001 | = FCK / 2 |
| 010 | = FCK / 4 |
| 011 | = FCK / 8 |
| 100 | = FCK / 16 |
| 101 | = FCK / 64 |
| 110 | = FCK / 256 |
| 111 | = FCK / 1024 |

13.3.3 P_CMT0_TCNT (0x7508) : Compare Match Timer 0 Counter Register

13.3.4 P_CMT1_TCNT (0x7509) : Compare Match Timer 1 Counter Register

Compare match timer counter is a 16-bit register used as an up-counter. The initial value is 0x0000.

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

13.3.5 P_CMT0_TPR (0x7510) : Compare Match Timer 0 Period Register

13.3.6 P_CMT1_TPR (0x7511) : Compare Match Timer 1 Period Register

The compare match timer period register is a 16-bit register used to set the period for compare match function. The initial value is 0x0000. The P_CMTx_TCNT (x = 0, 1) will be cleared to 0x0000 when a new value has been written to P_CMTx_TPR (x = 0, 1).

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

13.4 Design Tips

【Example 14-1】Set two channels compare match timer with frequency 1000 Hz and 5 Hz for CMT0 and CMT1 respectively.

```
void Init_CMT (void)
{
    P_CMT0_TPR->W = P_CMT1_TPR->W = 0x0000;
    P_CMT_Start->W = 0x0000;           /* stop all compare match timer */
    P_CMT_Ctrl->W = 0x0000;           /* reset setting */

    P_CMT_Ctrl->B.CKA = 0;            /* CMT0 clock select = FCK / 1 */
    P_CMT_Ctrl->B.CM0IE = 1;          /* CMT0 compare match interrupt enable */
    P_CMT_Ctrl->B.CKB = 7;            /* CMT1 clock select = FCK / 1024 */
```

```
P_CMT_Ctrl->B.CM1IE = 1;          /* CMT1 compare match interrupt enable */
P_CMT0_TPR->W = 19999;           /* CMT0 period for 1000 Hz at FCK = 20.0 MHz */
P_CMT1_TPR->W = 3905;            /* CMT1 period for 5 Hz at FCK = 20.0 MHz */

P_CMT_Start->B.ST0 = 1;          /* start CMT0 counter */
P_CMT_Start->B.ST1 = 1;          /* start CMT1 counter */
}

/***********************/

void IRQ7(void) __attribute__ ((ISR));
void IRQ7(void)
{
    if(P_INT_Status->B.CMTIF)
    {
        if(P_CMT_Ctrl->B.CM0IF)
        {
            P_CMT_Ctrl->B.CM0IF = 1; /* clear CM0IF flag */
            // place your codes here.
        }

        if(P_CMT_Ctrl->B.CM1IF)
        {
            P_CMT_Ctrl->B.CM1IF = 1; /* clear CM1IF flag */
            // place your codes here.
        }
    }
}
```

Listing 14-1 compare match timer design tips

14 Time Base Module and Buzzer Unit

14.1 Introduction

The Time Base Module is used to produce the reference clock needed by other modules on the chip. It comprises a 16-bit ripple counter, can generate reference clocks from FCK/2 ~ FCK/65536 in which supply the peripherals of SPMC75X family MCU chip. Time base counter can be cleared by writing 0x5555 to Time Base Reset Register (P_TMB_Reset).

By using the divider of the Time Base Module, a 50% duty cycle pulse can be produced to drive the buzzer device. The selected time base clock is sent to pin IOC4/BZO. Figure 14-1 shows the time-base and buzzer output timing.

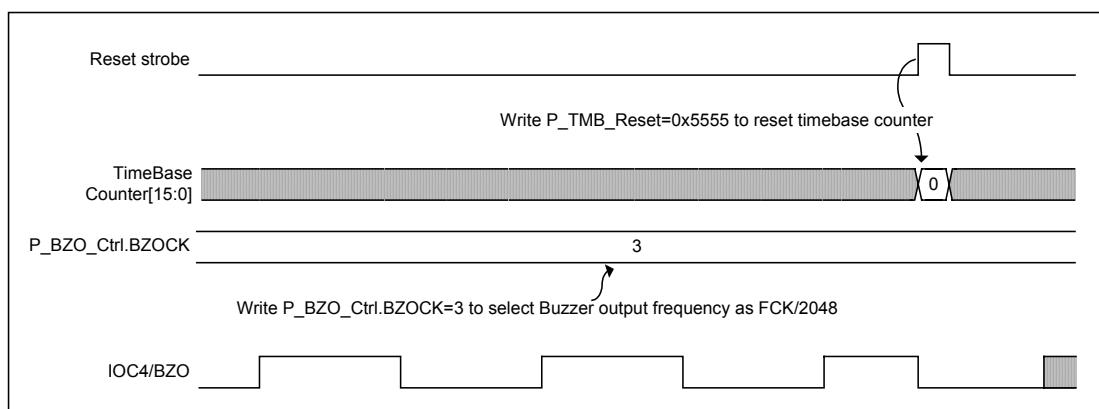


Figure 14-1 Time-Base and buzzer output timing

14.2 Timer Base Module and Buzzer Registers Address Tables

Table 15-1 Time base and buzzer registers

Address	Register	Name
70B8h	P_TMB_Reset	Time base reset register
70B9h	P_BZO_Ctrl	Buzzer output control register

14.3 Control Registers

14.3.1 P_TMB_Reset (0x70B8) : Time Base Reset Register

Write 0x5555h to this register to reset the time base counter register to initial the clock sources of all peripherals on the chip.

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

14.3.2 P_BZO_Ctrl (0x70B9) : Buzzer Output Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
BZOEN	Reserved						

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
Reserved							BZOCK

Bit 15: **BZOEN**, Buzzer output enable select bit.

0 = Disabled

1 = Enabled

Bit14:2: Reserved

Bit 1:0: **BZOCK**, Buzzer ouput frequency select bits

00 = FCK / 16384

01 = FCK / 8192

10 = FCK / 4096

11 = FCK / 2048

The following table summarizes the buzzer clock frequency with FCK = 24.0 MHz.

Table15-2 Buzzer output frequency BZOCK table

BZOCK	Buzzer output frequency	When FCK = 24MHz
00	FCK / 16384	1.465 kHz
01	FCK / 8192	2.929 kHz
10	FCK / 4096	5.859 kHz
11	FCK / 2048	11.718 kHz

14.4 Design Tips

【Example 15-1】: Setup buzzer unit with output clock frequency is 1500 Hz @ FCK = 24.0 MHz.

```
P_BZO_Ctrl->W = 0x0000;           /* reset Buzzer unit, and disable it */
P_BZO_Ctrl->B.BZOCK = 0;           /* buzzer clock frequency = FCK / 16384 */
P_BZO_Ctrl->B.BZOEN = 1;           /* enable buzzer clock output */
```

Listing 15-1 buzzer output control setting

15 Standard Peripheral Interface, SPI

15.1 Introduction

The SPMC75X family MCU has a three-pin SPI module as shown in Figure 16-1. The SPI is a high-speed synchronous serial I/O that allows a serial bit stream to be transmitted out or received into the device at a programmable transfer rate. The SPI supports full-duplex synchronous transfer between a master device and a slave device. The SPMC75X family MCU supports both master and slave modes. The parameters such as operation mode, clock frequency, clock phase, and clock polarity are user programmable. The SPI module provides the following features

- Three external pins:
 - SCK: clock input/output pin (shared with IOB11)
 - SDO: data output pin (shared with IOB13)
 - SDI: data input pin (shared with IOB12)
- Supports full-duplex synchronous transfer
- Two operation modes: master and slave
- Baud rate: 6 programmable transfer rate / Max. 6Mbps at 24MHz CPU clock
- Data word length: 8-bit
- Programmable clock phase and clock polarity settings
- Selectable data strobe time: input data bit sampled at the middle/end of data output time
- Three selectable sampling clock sources for noise immunity

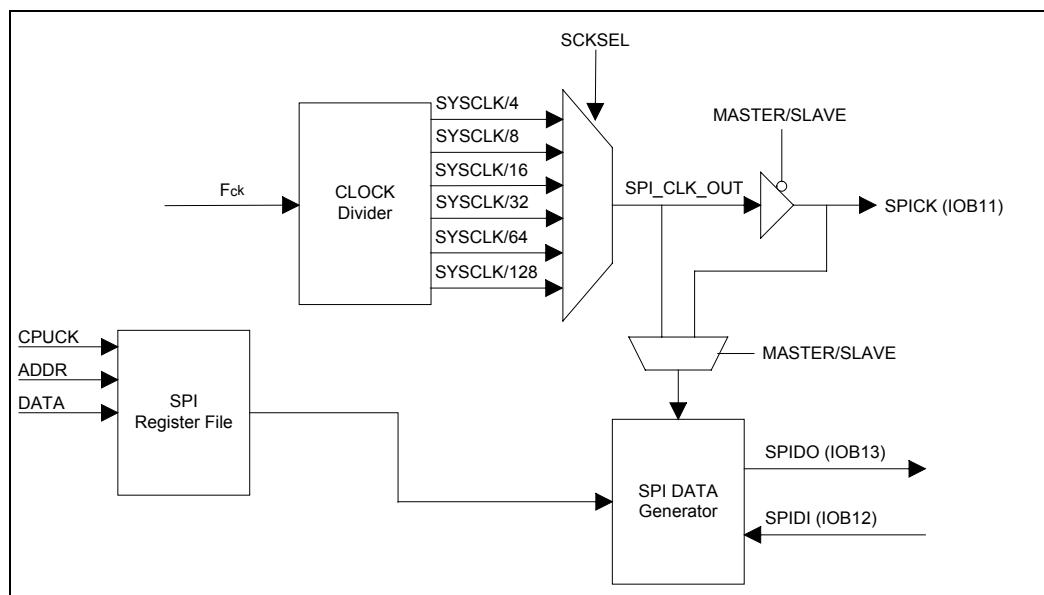


Figure 15-1 SPI structure

Following is a function diagram of SPI module

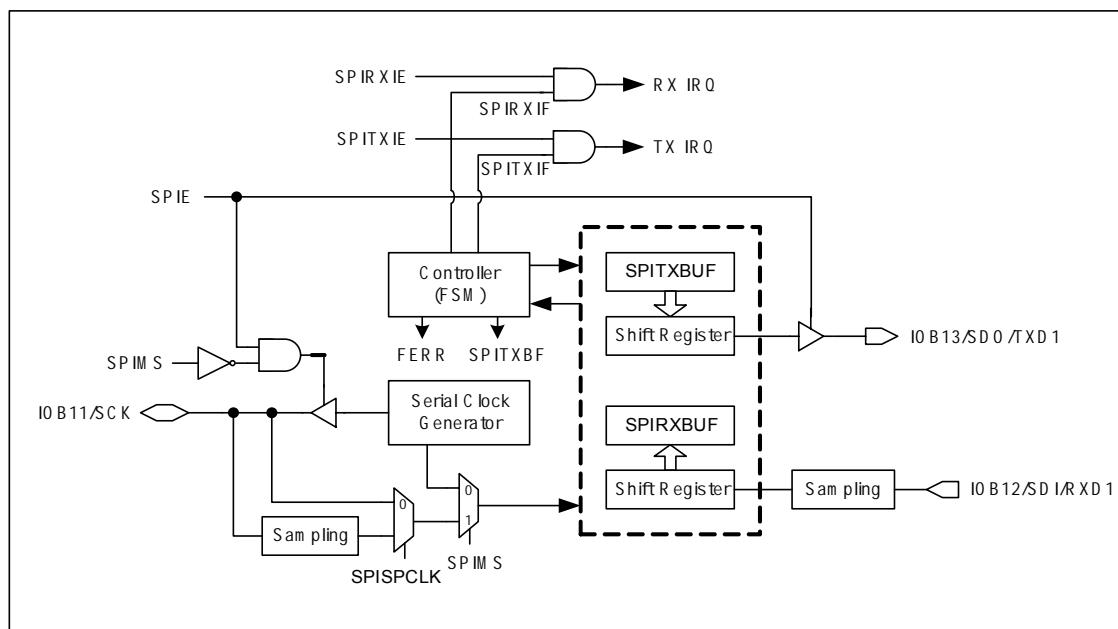


Figure 15-2 function block diagram of SPI interface

15.2 SPI Control Pins Configuration

Name	I/O	Description
SPICLK	I/O*	Serial Peripheral Interface, Clock Pin (Shared with IOB.11)
SPIDO	O	Serial Peripheral Interface, Data Out Pin (Shared with IOB.13)
SPIDI	I	Serial Peripheral Interface, Data In Pin (Shared with IOB.12)

* SPICLK is output when SPI operates master mode and SPICLK is input when SPI is setup as slave mode.

15.3 SPI Registers Address Table

Table 16-1 SPI registers

Address	Register	Name
7140h	P_SPI_Ctrl	SPI control register
7141h	P_SPI_TxStatus	SPI transmit status register
7142h	P_SPI_TxBuf	SPI transmission buffer register
7143h	P_SPI_RxStatus	SPI receive status register
7144h	P_SPI_RxBuf	SPI reception buffer register

15.4 Control Register

15.4.1 P_SPI_Ctrl (0x7140): SPI Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIE	Reserved			SPIRST	SPISPCLK	SPIMS	
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved		SPIPHA	SPIPOL	SPISMPS	SPIFS		

Bit 15 **SPIE**: SPI enable, If this bit is set to “1”, IOB[13:11] become SPI Interface. These pins can't be used as GPIO.

0 = Disable

1 = Enable

Bit 14:12 Reserved

Bit 11 **SPIRST**: Write 1 to reset. It only generate one pulse to reset the SPI module except for the register setting

Bit 10:9 **SPISPCLK**: Sampling clock select bits. Sampling clock is used for noise immunity

00 = no sampling

01 = FCK

10 = FCK/2

11 = FCK/4

Bit 8 **SPIMS**: SPI mode selection

0 = Master mode

1 = Slave mode

Bit 7:6 Reserved

Bit 5 **SPIPHA**: SPI clock phase. SPI clock phase select, see SPI Master Mode Timing.

Bit 4 **SPIPOL**: SPI clock polarity. SPI clock polarity select, ,see SPI Master Mode Timing.

Bit 3 **SPISMPS**: SPI sample mode selection for master mode

0 = input data bit sampled at the middle of data output time

1 = input data bit sampled at the end of data output time

Bit 2:0 **SPIFS**: Master mode clock frequency selection

000 = FCK/4

001 = FCK/8

010 = FCK/16

011 = FCK/32

100 = FCK/64

101, 110, 111 = FCK/128

15.4.2 P_SPI_TxStatus (0x7141): SPI Transmit Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R	R	R	R
0	0	0	0	0	0	0	0
SPITXIF	SPITXIE	SPITXBF	Reserved				

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Bit 15 **SPITXIF:** SPI Transmit interrupt flag

0 = Not occur

1 = Happened, write 1 to clear

Bit 14 **SPITXIE:** SPI Transmit interrupt enable

0 = Disable

1 = Enable

Bit 13 **SPITXBF:** Transmission buffer full flag. This bit is set to 1 by the hardware when the transmission buffer is full, it is cleared to 0 when the byte written to the P_SPI_TxBuf register has loaded to the shift register. The software should not update P_SPI_TxBuf register when SPITXBF bit is set, otherwise the new data will overwrite the old data to be send.

1 = Transmission buffer full

0 = Transmission buffer is empty

Bit 12:0 Reserved

15.4.3 P_SPI_TxBuf (0x7142): SPI Transmission Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPITXBUF							

Bit 15:8 Reserved

Bit 7:0 **SPITXBUF** = Write data sends to SPIDO pin

15.4.4 P_SPI_RxStatus 0x(7143): SPI Receive Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R	R
0	0	0	0	0	0	0	0
SPIRXIF	SPIRXIE	Reserved			FERR	Reserved	
B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

Bit 15 **SPIRXIF:** SPI receive interrupt flag

0 = Not occur

1 = Happened, write 1 to clear

Bit 14 **SPIRXIE:** SPI receive interrupt enable

0 = Disable

1 = Enable

Bit 13:11 Reserved

Bit 10 **FERR:** Buffer full and overwrite error bit.

0 = No overwrite error occurs

1 = overwrite error happened.

Bit 9:0 Reserved

15.4.5 P_SPI_RxBuf (0x7144): SPI Reception Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							
B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIRXBUF							

Bit 15:8 Reserved

Bit 7:0 **SPIRXBUF** = Read data from SPIDI pin

15.5 SPI Operation

15.5.1 Master Mode

As in master mode, the shifting clock (SPICLK) is generated by SPI module. There are two control bits to control the clock phase (SPIPH) and polarity (SIPOL) in the P_SPI_Ctrl register. The transmission starts immediately from data is written to the P_SPI_TxBuf register.

After software writes one byte through P_SPI_TxBuf register, the data is latched into its internal transmission buffer. If the shift register is not shifting data, the data will be loaded to the shift register and start transmitting at the next SCLK phase. On the other hand, if the shift register is busy in shifting data (SPITXBF flag is set in P_SPI_TxStatus register), the new data will not be shifted out until the present byte has been shifted out.

The SPI shifts the data from MSB to LSB through the SPIDO pin. The 8-bit data is shifted out after eight SCLK cycles. At the same time, the data is also shifted in through SDI pin. When each 8-bit transfer is completed, the SPITXIF bit in P_SPI_TxStatus register will be set; besides, a SPI interrupt will be generated if the SPITXIE bit is set to ‘1’ in P_SPI_TxStatus register.

In contrast, while SPI interface is received one byte successfully, the received data will be latched into reception buffer. At that time, SPIRXIF bit in P_SPI_RxStatus register will be set and a SPI interrupt will be issued to CPU if the SPIRXIE bit in the P_SPI_RxStatus register is set.

The following diagram depicts the timing scheme on SPI master mode for different operation types (polarity control bit equals “1” or “0”, phase control bit equals “1” or “0”, and sample strobe control bit equals “1” or “0”).

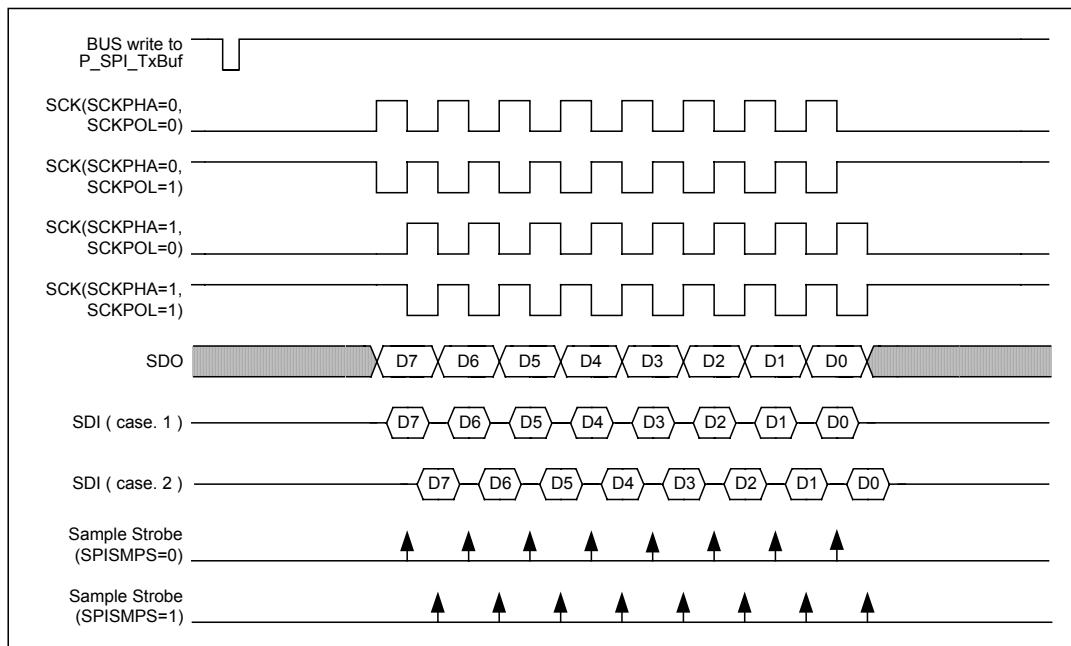


Figure 15-3 SPI mode timing, Master Mode

15.5.2 Slave Mode

In slave mode, the shifting clock SCLK comes from external SPI master, so the transmission starts from the first external SCLK event. To transmit, the firmware should write the data to its transmitting buffer before the first SCK comes from the master. Both master and slave devices must be programmed with the same SCLK phase and polarity for transmitting and receiving data.

If the clock phase bit (SPIPHA) is “1”, the first data bit to be shifted out starts right after the command written to P_SPI_TxBUF register. If the clock phase bit (SPIPHA) is “0”, the first data bit to be shifted will start after first SCLK edge.

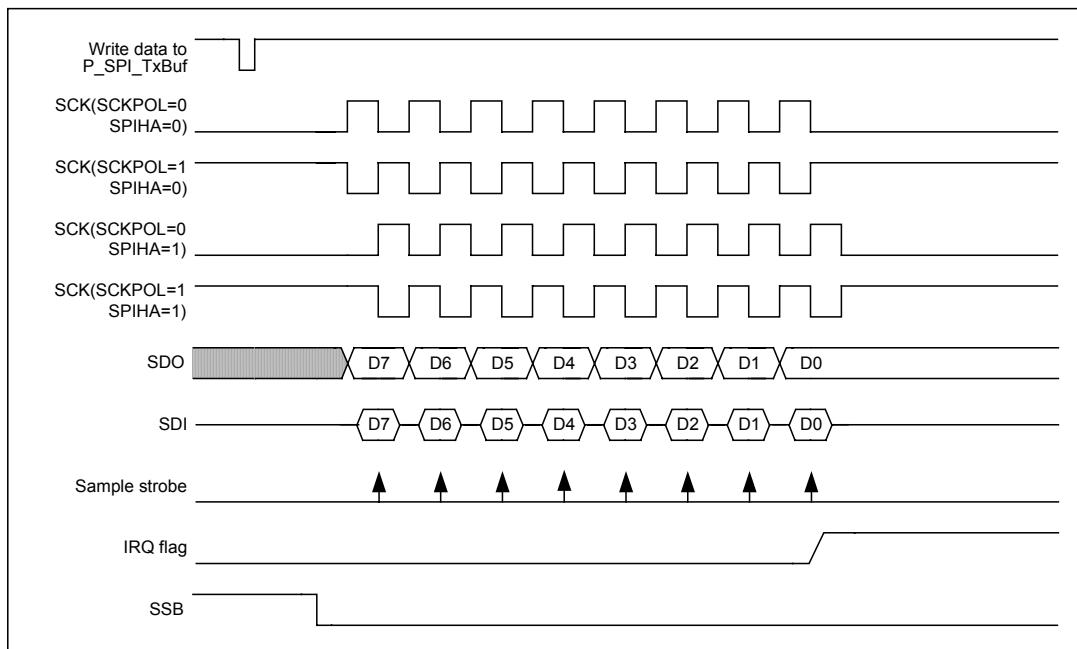


Figure 15-4 SPI mode timing, Slave Mode, SPIPHA = 0

15.6 Design Tips

【Example 16-1】 Enable SPI as master mode and transmite data.

```

P_SPI_Ctrl->B.SPIRST = CB_SPI_SPIRST;                                /* SPI Reset */
P_SPI_Ctrl->W  = CW_SPI_SPIE + CW_SPI_SPISPCLK_FCKdiv4 +
                  CW_SPI_SPIFS_CPUCLKdiv64;                         /* SPI Enable */
Data  = 0x0000;
while(1)
{
    P_SPI_TxBuf ->W  = Data;
    while(!P_SPI_TxStatus->B.SPITXIF);
    P_SPI_TxStatus->B.SPITXIF = CB_SPI_SPITXIF; /* Clear SPI Interrupt flag */
    F_Delay(10);                               /* delay about 10.0 ms for debugging use */
    Data++;
}
/* end while(1) */

```

Listing 16-1 SPI transmission with polling

【Example 14-1】 Enable SPI as master mode and transmite data with IRQ6.

```

P_SPI_Ctrl ->B.SPIRST = CB_SPI_SPIRST;                                /* SPI Reset */
P_SPI_Ctrl ->W  = CW_SPI_SPIE + CW_SPI_SPISPCLK_FCKdiv4 +
                  CW_SPI_SPIFS_CPUCLKdiv64;                         /* SPI Enable */
P_SPI_TxStatus ->B.SPITXIE = CB_SPI_SPITXIE; /* Enable SPI Interrupt */

IRQ_ON();                                /* enable IRQ */

Data = 0x0000;

P_SPI_TxBuf-> W = Data;

```

```
while(1);

/***********************/

void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{
    if(P_SPI_TxStatus->B.SPITXIF)           /* Check the SPI interrupt flag */
    {
        P_SPI_TxStatus->B.SPITXIF = CB_SPI_SPITXIF; /* Clear SPI interrupt flag */
        Data++;
        P_SPI_TxBuf -> W = Data;
    } /* end if */
}
```

Listing 16-2 SPI transmission with interrupt

16 Universal Asynchronous Receiver/Transmitter – UART

16.1 Introduction

The UART module built in SPMC75X family MCU performs serial-to-parallel conversion on data received from an external device and it performs parallel-to-serial conversion on data transmitted to the external device.. This module provides the following features.

- Provides standard asynchronous, full-duplex communication
- Programmable trans-receive baud rate
- Parity can be even, odd or disabled for generation and detection
- Stop bit width can be 1 or 2 bits
- Support transmitting interrupt
- Support receiving interrupt
- High noise rejection for bit receiving (majority decision of 3 consecutive samples in the middle of received bit time)
- Framing, Parity error detection during reception.
- Overrun detection
- programmable baudrate from 300 bps to 115200 bps
- Support Transmission/Reception data channel selection between TXD1/RXD1 and TXD2/RXD2.
- Transmission/Reception interface can be activated independently.

Figure 17-1 shows the UART block diagram of SPMC75X family.

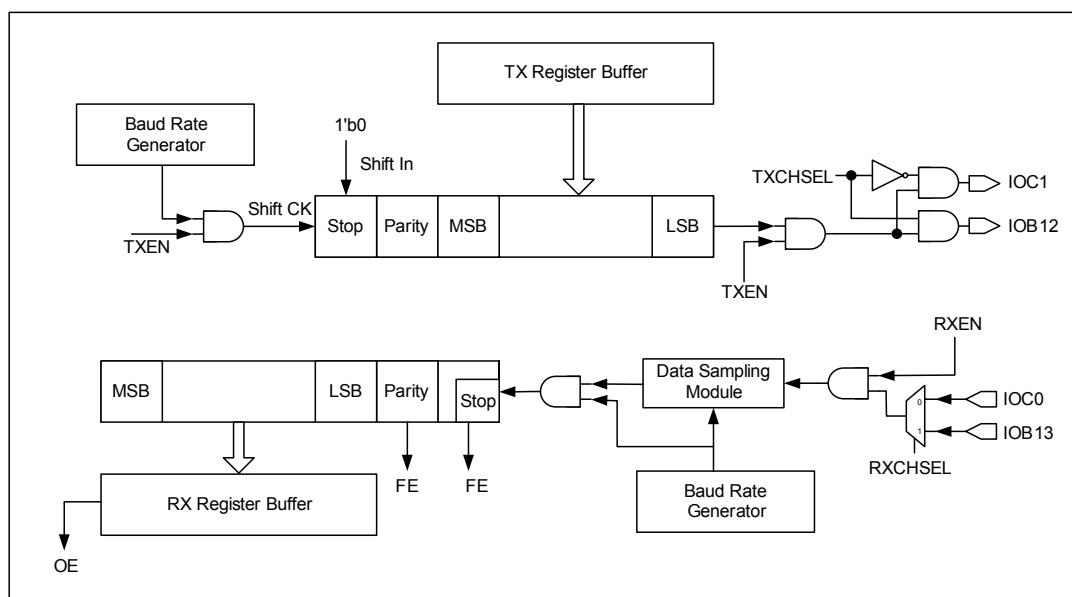


Figure 17-1 UART block diagram

16.2 Application Circuit

The UART interface normal used to communicate with PC computer. The following Figure 17-2 show the application circuit between PC and SPMC75X family MCU by using ICL232 chip.

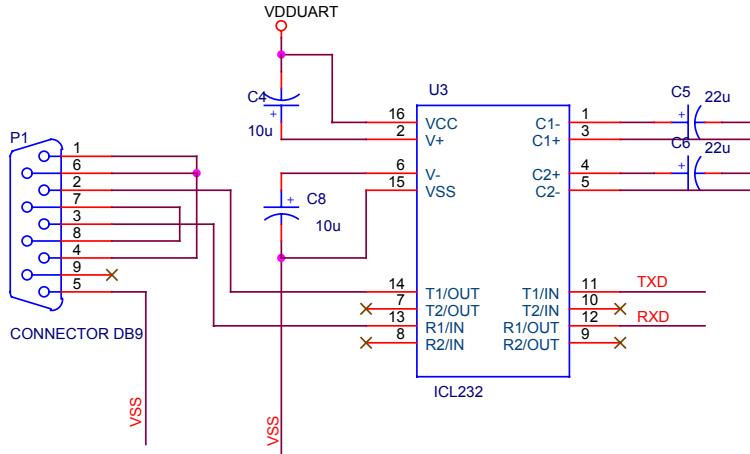


Figure 16-2 UART interface to host application circuit

16.2.1 UART Frame Scheme

UART data format is depicted in the following diagram.

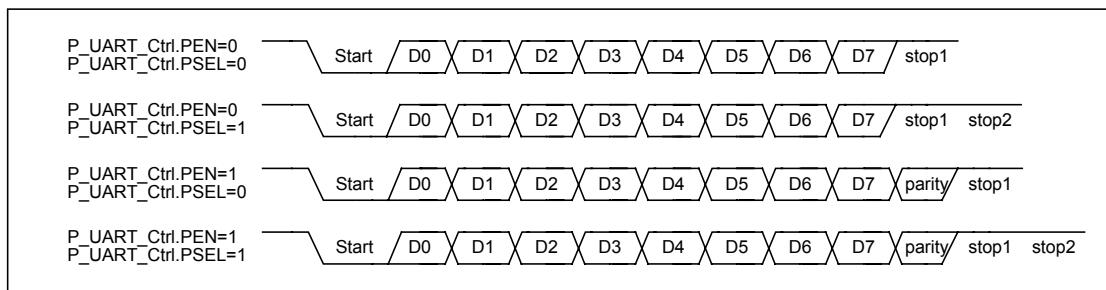


Figure 16-3 UART data format

16.3 UART Control Pin Configuration

Name	I/O	Description
RXD1	I	UART Reception Pin (shared with IOB12)
TXD1	O	UART Transmission Pin (shared with IOB13)
RXD2	I	UART Reception Pin (shared with IOC0)
TXD2	O	UART Transmission Pin (shared with IOC1)

16.4 UART Registers Address Table

Table 17-1 UART registers

Address	Register	Name
7100h	P_UART_Data	UART data register
7101h	P_UART_RxStatus	UART reception error flag register
7102h	P_UART_Ctrl	UART control register
7103h	P_UART_BaudRate	UART baud-rate setup register
7104h	P_UART_Status	UART status register

16.5 Control Register

16.5.1 P_UART_Data (0x7100): UART Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved			OE	Reserved		PE	FE
B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTDATA							

Bit 15:12 Reserved

Bit 11 **OE**: Overrun Error (Ready-only). the same as P_UART_RXStatus bit 3 This bit is set to 1 if data is received and the FIFO is full.

0 = Not Occurred

1 = Occurred

Bit 10 Reserved

Bit 9 **PE**: Parity Error (Ready-only). the same as P_UART_RXStatus bit 1 This bit is set to “1” if the parity of the received data character does not match the parity selected in PSEL Control bit.

0 = Not Occurred

1 = Occurred

Bit 8 **FE**: Frame Error (Ready-only). the same as P_UART_RXStatus bit 0 This bit is set to “1” if the received character does not have a valid stop bit (a valid stop bit is 1 bit).

0 = Not Occurred

1 = Occurred

Bit 7:0 **UARTDATA**: UART Data Read/Write Register.

This control register represents the entry to transmitting/receiving buffer.

Note: Read-only error flags in bit [11:8] have the same function with control register bits located in bit[3..0] of P_UART_RXStatus register.

16.5.2 P_UART_RXStatus (0x7101): UART Reception Error Flag Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	Bb2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved				OE	Reserved	PE	FE

Bit 15:4 Reserved

Bit 3 **OE**: Overrun Error. This bit is set to “1” if data is received and the buffer is full.

Read 0 = Not Occurred

Read 1 = Occurred

Bit 2 Reserved

Bit 1 **PE**: Parity Error. This bit is set to “1” if the parity of the received data character does not match the parity selected in PSEL Control bit.

Read 0 = Not Occurred

Read 1 = Occurred

Bit 0 **FE**: Frame Error. This bit is set to “1” if the received character does not have a valid stop bit (a valid stop bit is 1 bit).

Read 0 = Not Occurred

Read 1 = Occurred

16.5.3 P_UART_Ctrl (0x7102): UART Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	W	R/W	R/W	R
0	0	0	0	0	0	0	0
RXIE	TXIE	RXEN	TXEN	Reset	TXCHSEL	RXCHSEL	Reserved

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
Reserved				SBSEL	PSEL	PEN	Reserved

Bit 15 **RXIE**: Receive Interrupt Enable. If this bit is set to “1”, and the receive buffer is just held one new-coming data, hardware will issue an IRQ6 or FIQ to CPU.

0 = Disabled

1 = Enabled

Bit 14 **TXIE**: Transmit Interrupt Enable. If this bit is set to “1”, and the transmit buffer is empty, hardware will issue an IRQ6 or FIQ to CPU.

0 = Disabled

1 = Enabled

Bit 13 **RXEN**: UART reception enable, If this bit is set to “1”, the UART reception interface is enabled, RXD1/RXD2 is input floating with programming the RXCHSEL bit in P_UART_Ctrl register.

0 = Disabled

1 = Enabled

Bit 12 **TXEN**: UART transmission enable, If this bit is set to “1”, the UART transmission interface is enabled, TXD1/TXD2 is buffer output with programming the TXCHSEL bit in P_UART_Ctrl register.

0 = Disabled

1 = Enabled

Bit 11 **Reset**: Software reset.

Bit 10 **TXCHSEL**: Transmission data channel selection

0 = UART transmission to TXD2 on IOC1 pin

1 = UART transmission to TXD1 on IOB12 pin

Bit 9 **RXCHSEL**: Reception data channel selection

0 = UART reception from RXD2 on IOC0 pin

1 = UART reception from RXD1 on IOB13 pin

Bit 8:4 Reserved

Bit 3 **SBSEL**: Stop Bit Size Selection.

When this bit is set to “1”, two stop bits are transmitted at the end of the frame. The receiving logic does not check for two stop bits being received.

0 = 1 Stop Bit

1 = 2 Stop Bit

Bit 2 **PSEL**: Parity Selection.

If this bit is set to “1”, even parity generation and checking are performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to “0” the odd parity is performed to check for an odd number of 1s. This bit has no effect when parity is disabled by PEN Control bit is cleared to “0”.

0 = Odd Parity (if PEN= 1)

1 = Even Parity (if PEN= 1)

Bit 1 **PEN**: Parity Enable.

If this bit is set to “1”, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.

0 = Disabled

1 = Enabled

Bit 0 Reserved

16.5.4 P_UART_BaudRate(0x7103): UART Baud Rate Setup Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

Bit 15:0 **UARTBUD**: UART Baud Rate Divisor.

$$\text{Baud Rate} = \text{FCK} / [16 \times (65536 - \text{P_UART_BaudRate})]$$

The value of P_UART_BaudRate register is calculated as follows :

$$\text{P_UART_BaudRate} = 65536 - \text{FCK} / (16 \times \text{Baud Rate})$$

16.5.5 P_UART_Status (0x7104): UART Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	1	0	0	0	0	0	0
RXIF	TXIF	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved	RXBF	Reserved		BY	Reserved		

Bit 15 **RXIF**: Receive Interrupt Flag. This flag indicates a new data is received complete and issue UART receive interrupt. The RXIF bit will be automatically cleared to '0' if data is read out from P_UART_Data register or disabling reception by setting RXEN to '0'.

1 = a valid byte received complete, an interrupt is asserted if RXIE bit is set as '1'

0 = no reception interrupt

Bit 14 **TXIF**: Transmit Interrupt Flag. If the transmit buffer is ready to accept new data, this flag is set '1' by hardware and issue UART transmit interrupt. The TXIF bit will be automatically cleared to '0' if a new data is written to P_UART_Data register or disabling transmission by setting TXEN to '0'..

1 = transmitter is ready, an interrupt is asserted if TXIE bit is set as '1'

0 = transmitter is not ready

Bit 13:7 Reserved

Bit 6 **RXBF**: receiving buffer full flag . This flag is read-only, hardware will set or clear this flag automatically.

0 = reception buffer is not full
1 = reception buffer is full

Bit 5:4 Reserved

Bit 3 **BY:** transmitting busy flag.
If this bit is read as “1”, the UART module is busy in transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. Note that this flag is read-only, hardware will set or clear this flag automatically.

0 = transmitter is ready
1 = transmitter is busy

Bit 2:0 Reserved

16.6 UART Operation

The UART receive status is read from P_UART_RXStatus control register. The status information corresponds to the data character read from P_UART_Data control register prior to reading P_UART_RXStatus control register. Write “1” to corresponding register bit will clear the frame, parity break and overrun error.

Note that, the received data byte must be read first from P_UART_Data before reading the corresponding error status from P_UART_RXStatus. This read sequence cannot be reversed since the status register P_UART_RXStatus is updated only when a read operation is performed on P_UART_Data control register.

There exists a baud rate register and a 16-bit timer to generate the baud rate. Each times the timer increments from its maximum count (0xFFFF), a clock is sent to the baud rate circuit. The clock is through divid-by-16 counter to generate the baud rate. The timer is reloaded automatically the value in baud rate register.

$$\text{Baud Rate} = \text{FCK} / [16 \times (65536 - \text{P_UART_BaudRate})]$$

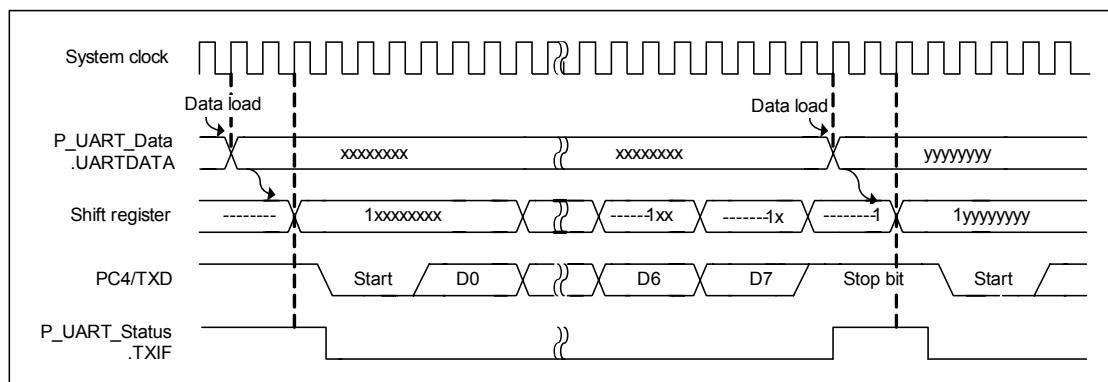
The content in baud rate register is taken as a 16-bit unsigned number. To derive the required baud rate register values from a known baud rate, use the equation:

$$\text{P_UART_BaudRate} = 65536 - \text{FCK} / (\text{16} \times \text{Baud Rate})$$

Table 17-2 P_UART_BaudRate setup value at FCK = 24.0 MHz

Baud Rate	Baud Rate Timer Reload Register Value @ 24MHz
115200 bps	0xFFFF3
57600 bps	0xFFE6
19200 bps	0xFFB2
9600 bps	0xFF64
4800 bps	0xFEC8
2400 bps	0xFD8F
1200 bps	0xFB1E
600 bps	0xF63C
300 bps	0xEC78

The UART begins transmitting after the first rollover of the divide-by-16 counter after the software writes to the P_UART_Data register. The UART transmits data on the TXD2/TXD1 pin in the following order: start bit, 8 data bits (LSB first), parity bit (Parity Enable mode only), stop bit. The TXIF bit in P_UART_Status register is set after 2 FCK cycles when the stop bit is transmitted. The TXIF bit is cleared automatically after the software writes to the P_UART_Data register and refer to Figure 17-4 for data transmission timing details..


Figure 17-4 Data transmit timing

Reception begins at the falling edge of a start bit received on RXD2/RXD1 pin, when enabled by the RXEN bit in P_UART_Ctrl register. For this purpose, RXD2/RXD1 is sampled 16 times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receiving clock is reset to align the counter rollover to the bit boundaries.

For noise rejection, the serial port establishes the content of each received bit by a majority decision of 3 consecutive samples in the middle of each bit time. This is especially true for the start bit. If the falling edge on RXD2/RXD1 is not verified by a majority decision of 3 consecutive samples logic low level, then the serial port stops reception and waits for another falling edge on RXD2/RXD1. Figure 17-5 shows the data sampling method.

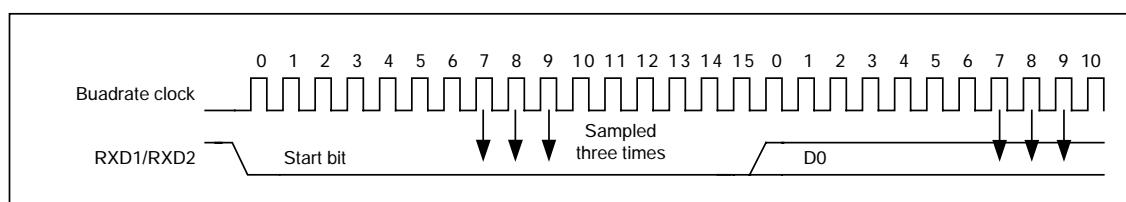


Figure 16-5 Data sampling method

After receiving the stop bit, the UART module writes the received byte to the P_UART_Data register and set the RXIF and RXBF bit. The serial port then waits for another high-to-low transition on the RXD1/RXD2 pin. Refer to Figure 17-6 for RXIF timing details.

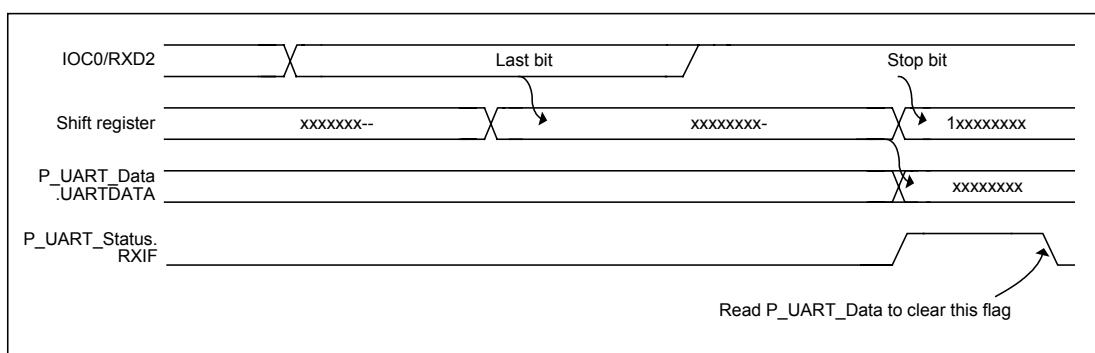


Figure 16-6 Receive buffer full RXIF timing

If the received byte is not read out before the next reception finished, the data will be over-written by the new received. In every reception session, RXBF is checked after receiving the stop bit. If the RXBF bit is set, the OE will be set to record this overrun error event. Remarkably, the OE will be cleared automatically if the error check success in the following session. Figure 17-7 indicates the OE flag timing.

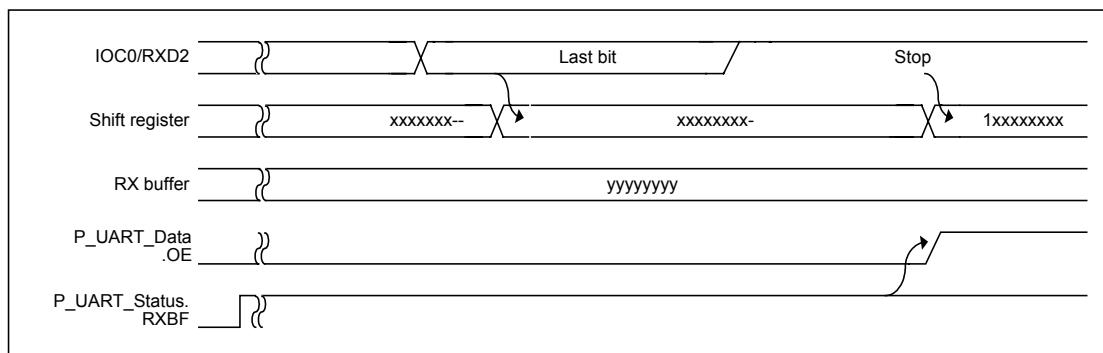


Figure 16-7 Receive buffer over-run OE timing

The parity and frame check is used for improving the reliability of reception. The parity can be even or odd according to the configuration of P_UART_Ctrl.PSEL. The parity check is performed after receiving parity bit if P_UART_Ctrl.PEN is enabled. The PE bit will be set if any parity error. Please refer to Figure 17-8 and Figure 17-9 for timing diagram. The Stop Bit is the part of the UART data formation. If the

reception session fails to receive Stop Bit, the integrity of the data frame is lost. The FE bit is set to record this frame error event. Remarkably, PE and FE will be clear automatically if the error checks success in the following session, respectively.

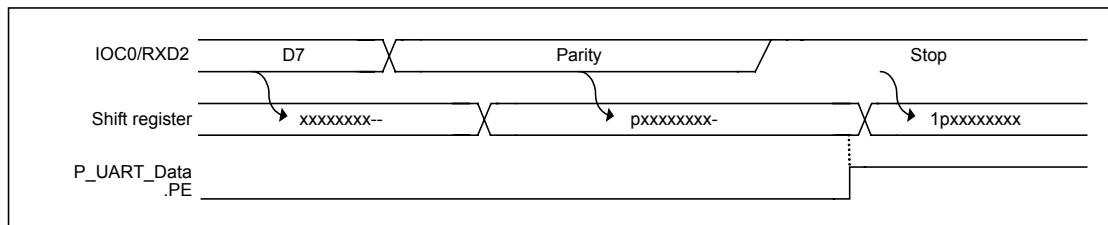


Figure 16-8 Parity error PE timing

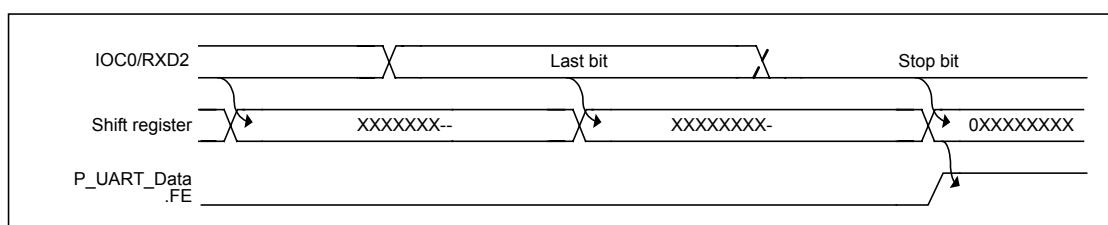


Figure 16-9 Frame error FE timing

16.7 Design Tips

【Example 17-1】 :Setup full-deplexing mode, interrupt driven flow for RXD2 and TxD2 of UART.

```
void UART_Test(void)
{
    unsigned int code = 0;

    P_UART_Ctrl->W = CW_UART_RXCHSEL_No2 | CW_UART_TXCHSEL_No2 | \
                      CW_UART_RXEN | CW_UART_TXEN | CW_UART_RXIE;
    /* transceiver pins on RXD2/TxD2, UART enable, enable RX interrupt */
    P_UART_BaudRate->W = 65406;           /* 9600 bps at 20 MHz cpu clock */
    P_UART_RXStatus->W = 0x000B;          /* clear OE, FE, PE error flag */
} /* UART_Test() */

/***********************/
*/
unsigned int txd_data = 0;
void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{
    unsigned int msg;
    if(P_INT_Status->B.UARTIF)
    {
        if(P_UART_Status->B.TXIF && P_UART_Ctrl->B.TXIE)
        {

```

```
// transmit data buffer to TXD2
P_UART_Data->W = txd_data;           /* demo code for transmission */
}
if(P_UART_Status->B.RXIF)
{
    // receive data from RXD2
    msg = P_UART_Data->W & 0xFF;      /* demo code for reception */
}
}
} /* IRQ6() */
```

Listing 17-1 UART interrupt setting

17 10-bit A/D Converter

17.1 Introduction

The SPMC75X family MCU embeds an 8-channel ADC with 10-bit resolution. The channel inputs AN7 – AN0 of ADC shares with GPIO pins IOA7 – IOA0, respectively. When corresponding ADC channel is enabled, each pin can be controlled to disable digital function through the register, P_ADC_Channel. The output of sample hold is converted from the analog signal fed into the converter. This converter generates a result via successive approximation. The analog top reference voltage is selectable through the pin VEXTREF. The A/D Converter used for the SPMC75X family MCU contains a sample hold circuit as illustrated below to fetch analog input voltage into the sample hold capacitor after activation A/D conversion. The block diagram of A/D converter is shown in Figure 18-1 and Figure 18-2 shows the timing diagram of ADC. The ADC has the following features:

- 10-bit resolution
- Max. 100kHz conversion rate
- 8 selectable input channels AN[7:0], shared pin with IOA[7:0]
- External reference input pin VEXTREF
- Four selectable ADC conversion clock: FCK/8, FCK/16, FCK/32, FCK/64
- Provides conversion complete interrupt
- Multiple triggers to start a conversion
 - Software immediate start
 - TGRA compare match on PDC0, PDC1, TPM2, MCP3, and MCP4 timers

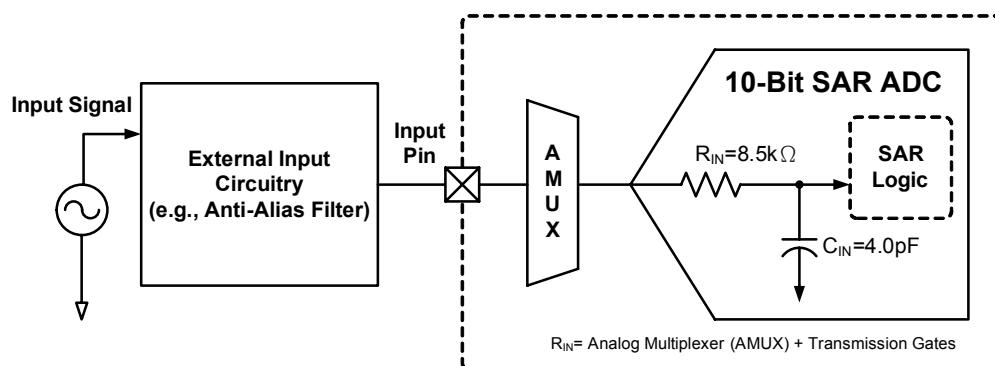


Figure 17-1 ADC function block

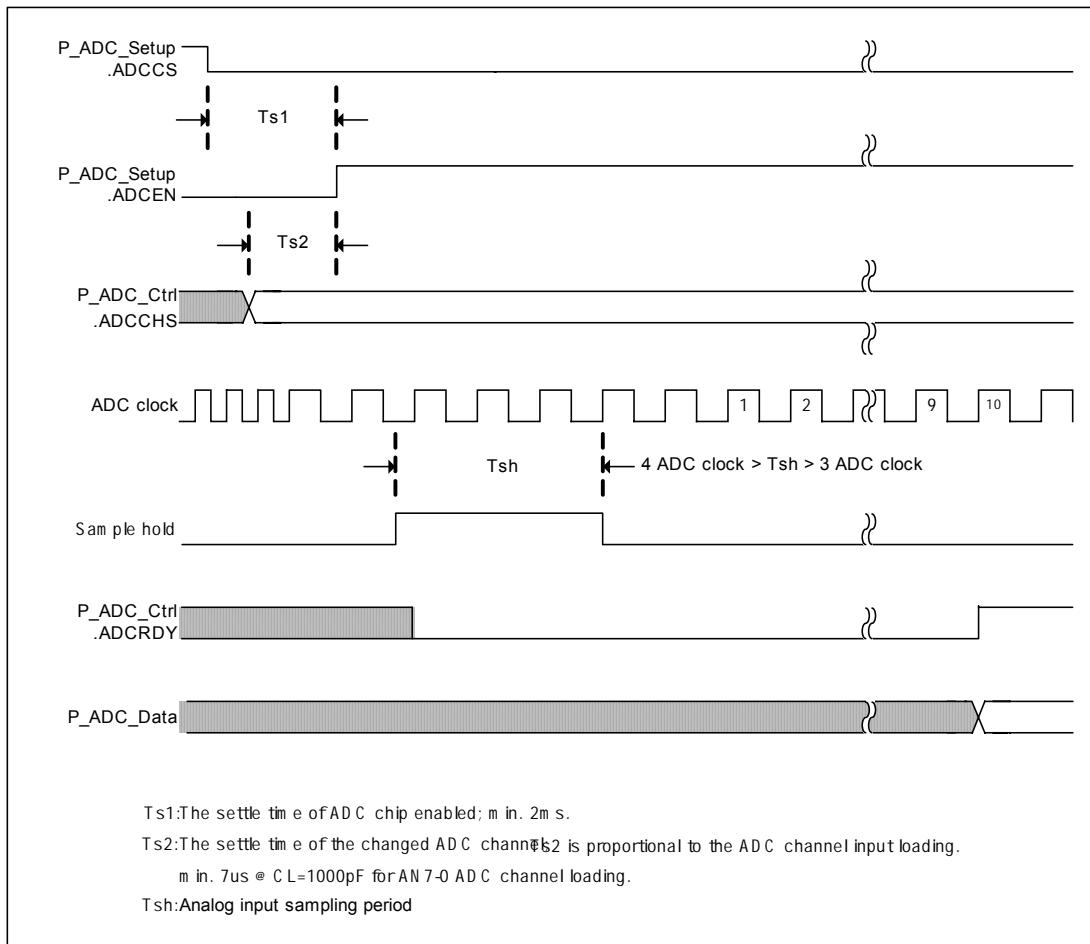


Figure 18-2 ADC timing diagram

17.2 ADC Registers Address Table

Table 18-1 ADC registers

Address	Register	Name
7160h	P_ADC_Setup	ADC setup register
7161h	P_ADC_Ctrl	ADC control register
7162h	P_ADC_Data	ADC data register
7163h	P_ADC_Channel	ADC input channel select register

17.3 Control Registers

The P_ADC_Setup register controls the ADC block power on or off, ADC conversion clock and event selection to trigger the start operation of ADC. User should note that when power-on-reset occurred, the ADC block is power on (ADCCS bit is 1) and ADC function is off (ADCEN bit is 0). At meanwhile, the P_ADC_Data value is 0xFFC0 for the purpose of power saving but without ADC conversion ready signal (ADCRDY in P_ADC_Ctrl register is 0). If user sets the ADCEN to 1 at this time, the ADC block will generate the ADCRDY signal and also set the ADCIF bit in P_ADC_Ctrl register. To prevent read the incorrect ADC value; do not read the first ADC data after the ADCEN is set to 1.

In SPMC75X family MCU, the ADC module conversion rate is limited to 1.5 MHz. User should note that do not set the ADC conversion frequency above this limitation, otherwise may get unstable ADC conversion results.

17.3.1 P_ADC_Setup (0x7160) : ADC Setup Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
1	0	0	0	0	0	0	0
ADCCS	ADCEN	Reserved			ADCFCS	ADCEXTRG	
B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ASPEN	Reserved						

Bit 15 **ADCCS**: ADC power on

- 0 = un-select ADC block
- 1 = select ADC block

Bit 14 **ADCEN**: ADC converter enable, When ADC is enabled, power consumption will increase rapidly. Therefore, turn ADC on only when necessary, and turn off ADC immediately after ADC data is successfully acquired

- 0 = disable ADC block
- 1 = enable ADC block

Bit13:11 Reserved

Bit 10:9 **ADCFCS**: A/D converter clock selection. Do not select FCK/8 when CPU operating speed is 24.0 MHz.

- 00 = FCK /8
- 01 = FCK /16
- 10 = FCK /32
- 11 = FCK /64

Bit 8 **ADCEXTRG**: External ADC conversion request trigger from a high pulse on IOA15 pad. IOA15 can be set as floating, pull-up and pull-down configuration

- 1 = enable
- 0 = disable

Bit 7 **ASPEN**: Auto Sampling mode enable, the sampling request signal is from PDC, TPM, MCP timer modules.

- 0 = Disable
- 1 = Enable

Bit 6:0 Reserved

17.3.2 P_ADC_Ctrl(0x7161) : ADC Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCIF	ADCIE	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ADCRDY	ADCSTR	Reserved			ADCCHS		

Bit 15 **ADCIF**: ADC interrupt flag. Status flag that indicates ADC have been converted complete. The flag will be cleared if '1' has been written to.

0 = interrupt Not happened

1 = interrupt happen

Bit 14 **ADCIE**: ADC interrupt enable. Enables or disables generation of ADC conversion interrupt.

0 = disable

1 = enable

Bit 13: 8 Reserved

Bit 7 **ADCRDY**: ADC conversion ready

0 = conversion not ready, AD data not effect

1 = conversion ready, AD data ist effect

Bit 6 **ADCSTR**: Manual Start ADC Conversion. Write this bit to "1" will start the operation of ADC conversion.

0 = No Effect

1 = START

Bit 5:3 Reserved

Bit 2:0 **ADCCHS**: Select ADC converter channel input

000 = ADC Channel0 (IOA0)

001 = ADC Channel1 (IOA1)

010 = ADC Channel2 (IOA2)

011 = ADC Channel3 (IOA3)

100 = ADC Channel4 (IOA4)

101 = ADC Channel5 (IOA5)

110 = ADC Channel6 (IOA6)

111 = ADC Channel7 (IOA7)

17.3.3 P_ADC_Channel(0x7166H) : ADC Input Channels Select Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W							
0	0	0	0	0	0	0	0
ADCCH7	ADCCH6	ADCCH5	ADCCH4	ADCCH3	ADCCH2	ADCCH1	ADCCH0

Bit 15: 8 Reserved

Bit 7 **ADCCH7:** ADC Input Channel 7 Enable

1 = IOA7 as ADC channel 7

0 = IOA7 as GPIO

Bit 6 **ADCCH6:** ADC Input Channel 6 Enable

1 = IOA6 as ADC channel 6

0 = IOA6 as GPIO

Bit 5 **ADCCH5:** ADC Input Channel 5 Enable

1 = IOA5 as ADC channel 5

0 = IOA5 as GPIO

Bit 4 **ADCCH4:** ADC Input Channel 4 Enable

1 = IOA4 as ADC channel 4

0 = IOA4 as GPIO

Bit 3 **ADCCH3:** ADC Input Channel 3 Enable

1 = IOA3 as ADC channel 3

0 = IOA3 as GPIO

Bit 2 **ADCCH2:** ADC Input Channel 2 Enable

1 = IOA2 as ADC channel 2

0 = IOA2 as GPIO

Bit 1 **ADCCH1:** ADC Input Channel 1 Enable

1 = IOA1 as ADC channel 1

0 = IOA1 as GPIO

Bit 0 **ADCCH0:** ADC Input Channel0 Enable

1 = IOA0 as ADC channel 0

0 = IOA0 as GPIO

17.3.4 P_ADC_Data (0x7162H) : ADC Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
1	1	1	1	1	1	1	1
ADCDATA							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
1	1	0	0	0	0	0	0
ADCDATA		Reserved					

Bit 15:6 **ADDATA**: ADC conversion data. After Control bit ADCRDY is set, the AD acquired data will be valid on this register

Bit 5:0 Reserved

17.4 ADC Operation

The bit ADCCS is set initially to turn on the internal bias in ADC and off if Standby mode for power saving. ADCEN is the control bit to enable ADC function. The ADC clock is configured by ADCFS in P_ADC_Setup. The derived clock frequency is suggested to be less 1.5MHz for conversion precision. The ways to starting conversion have tree methods: external conversion request, auto sampling signal from Timer/PWM module (TPM) and manual ADC conversion. These can be configured by the bits ADCEXTRG and ASPEN in P_ADC_Setup, and ADCSTR in P_ADC_Ctrl. The conversion ready status ADCRDY and interrupt flag ADCIF will set if conversion ready. The converted data can acquired through P_ADC_Data register. The example of ADC operation is shown in Figure 18-3.

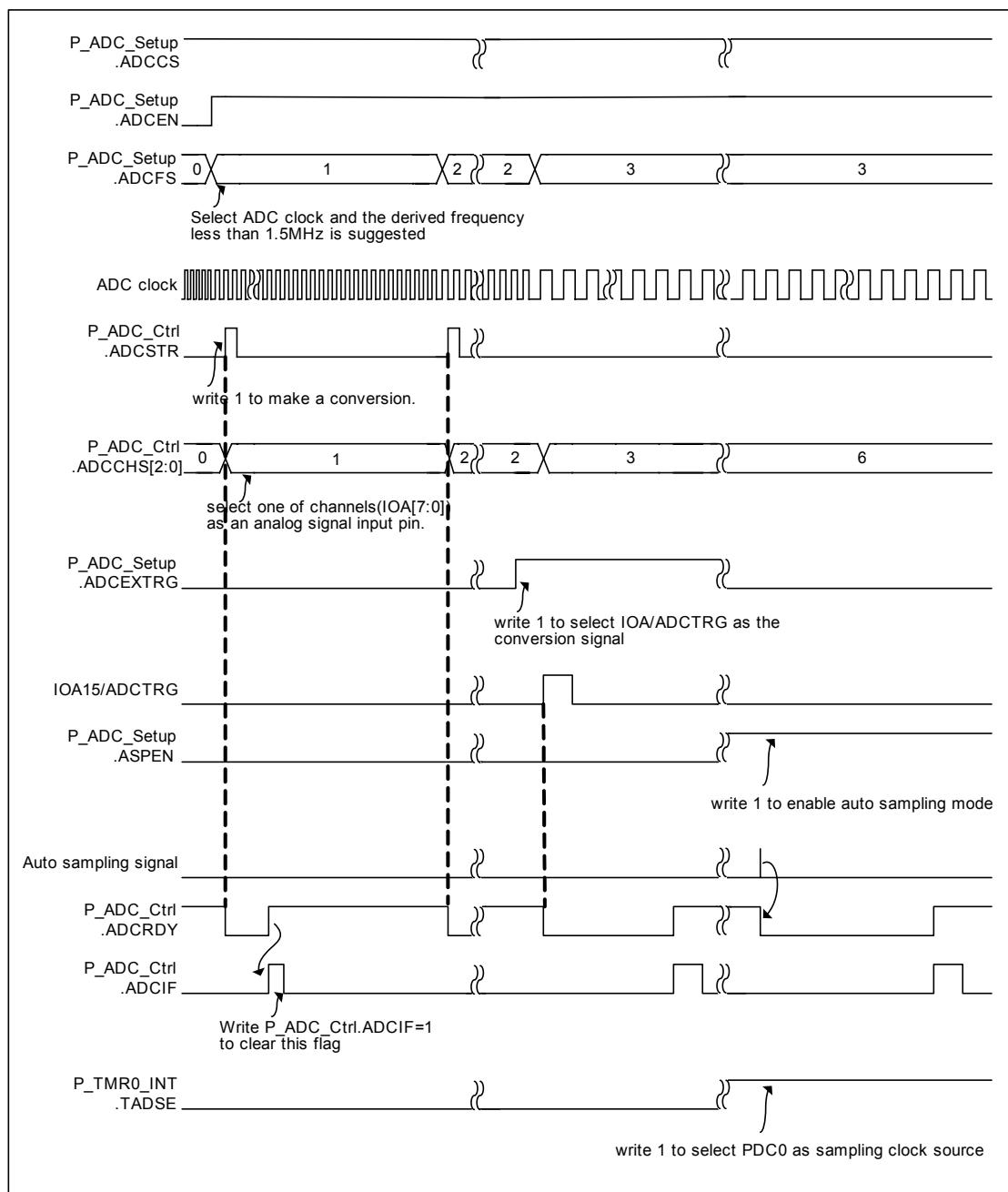


Figure 17-2 ADC conversion timing

17.5 Design Tips

[Example 18-1] Enable ADC and polling ADCRDY bit.

```

unsigned int Times;
unsigned int uiAdcData[8] = {0}; /* ADC result buffer */

P_ADC_Setup->W = CW_ADC_ADCCS_Select +
    CW_ADC_ADCEN + CW_ADC_ADCFS_CPUCLKdiv64;

```

```

/* ADC converter enable;A/D converter clock selection is CPUCLK/64 */
P_ADC_Channel->W = 0x00FF;                                /* ADC Input Channel 7~0 Enable */
P_ADC_Ctrl->W = CW_ADC_ADCSTR + CW_ADC_ADCCHS_Ch0;

/* Manual Start ADC Conversion. first Ch0 */
for(i = 0; i < 8; i++)
{
    while(!(P_ADC_Ctrl->B.ADCRDY & CB_ADC_ADCRDY));
                           /* Check the ADC ready */
    {
        uiAdcData[i] = (P_ADC_Data->W >> 6) & 0x03FF;
                           /* Low 10 bit is effective */
        P_ADC_Ctrl->B.ADCCHS = i;
                           /* Set next ADC channel */
        P_ADC_Ctrl->B.ADCSTR = CB_ADC_ADCSTR;      /* Manual Start ADC */
    } /* end while */
} /* end for */
P_ADC_Setup->B.ADCEN = ~CB_ADC_ADCEN;

```

Listing 18-1 ADC polling design tips

【Example 18-2】 Enable ADC and change channel with interrupt

```

unsigned int Times;
unsigned int uiAdcData[8] = {0};                         /* ADC result buffer */

P_ADC_Setup->W = CW_ADC_ADCCS_Select+
                  CW_ADC_ADCEN + CW_ADC_ADCFS_CPUCLKdiv64;
/* ADC converter enable;A/D converter clock selection is CPUCLK/64 */
P_ADC_Channel->W = 0x00FF;      /* ADC Input Channel 7~0 Enable */
Times = 0x0000;
P_ADC_Ctrl->W = CW_ADC_ADCIE + CW_ADC_ADCSTR + CW_ADC_ADCCHS_Ch0;
                           /* Manual Start ADC Conversion. first Ch0 */
INT_IRQ();                /* Enable interrupt */
while(Times<8);
P_ADC_Ctrl->B.ADCIE = ~CW_ADC_ADCIE;           //Disable ADC Interrupr
P_ADC_Setup->B.ADCEN = ~CB_ADC_ADCCS_Select;

/*********************************************
***,
extern unsigned int Times;
extern unsigned int uiAdcData[8];

void IRQ6(void) __attribute__ ((ISR));
void IRQ6(void)
{

```

```
if(P_ADC_Ctrl->B.ADCIF & CB_ADC_ADCIF) /* Check the ADC interrupt flag */
{
    while(!(P_ADC_Ctrl->B.ADCRDY & CB_ADC_ADCRDY)) /* Check the ADC ready */
        if(Times==8)      break;

    P_ADC_Ctrl->B.ADCIF = CB_ADC_ADCIF;           /* Clear ADC interrupt flag */
    uiAdcData[Times] = (P_ADC_Data->W >> 6) & 0x03FF;
                                              /* Low 10 bit is effective */
    P_ADC_Ctrl->B.ADCCHS = Times+1;               /* Set next ADC channel */
    P_ADC_Ctrl->B.ADCSTR = CB_ADC_ADCSTR;         /* Manual Start ADC */
    Times++;
} //end if
}
```

Listing 18-2 ADC interrupt design tips

18 WATCHDOG

18.1 Introduction

The purpose of watchdog is to monitor if the system operates normally. Within a certain period, watchdog counter must be cleared. If the watchdog is not cleared, CPU assumes the program has been running in an abnormal condition and therefore, CPU will reset the system to the initial state and start running the program all over again. It protects the system from incorrect code execution by launching a system reset when the watchdog timer overflows as a result of failure of software to clear the timer within selection timer. For SPMC75X family MCU , Watchdog function can be enabled or disabled by P_System_Option.WDG.

The device includes a watchdog timer (WDT) to monitor abnormal software run-away. A system or CPU reset will be generated if it is not periodically cleared by software. The WDT operates independently of the CPU. And the watchdog timer is an eight-bit counter, its clock can be selected from eight different sources. When a counter overflow occurs, a watchdog reset will be generated. A watchdog reset can issue a system reset or CPU reset according to control register settings. To further ensure the settings of watchdog control register will not be modified accidentally, a special bit pattern must be written to the unused bits of watchdog control register when the settings is to be changed. Otherwise a watchdog reset will be generated if the unused bits of watchdog control register are not properly written. Figure 19-1 shows the watchdog timer timing diagram.

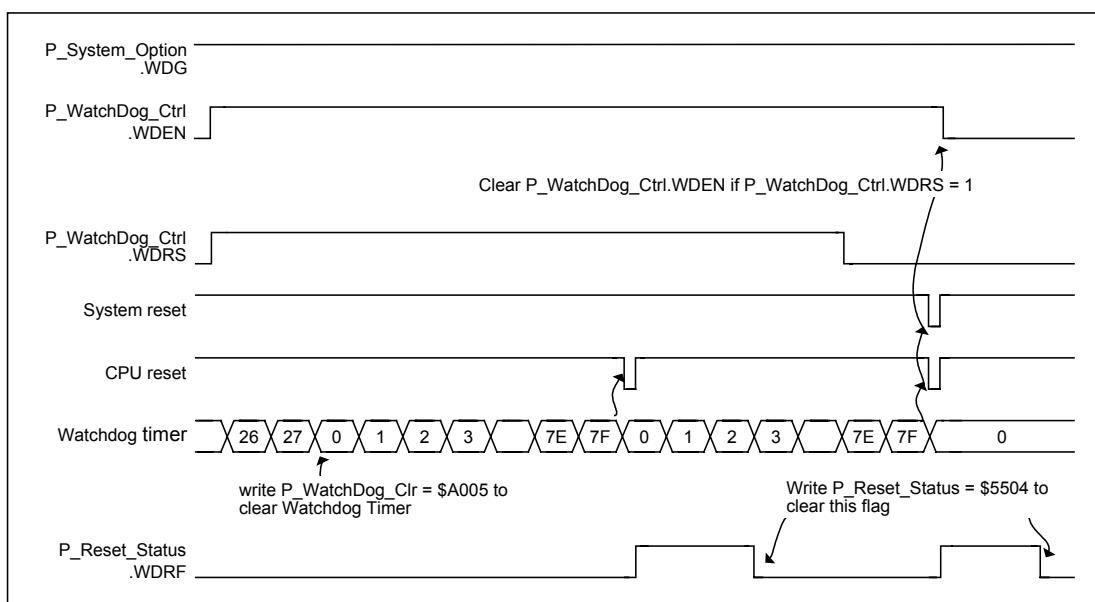


Figure 18-1 Watchdog timer timing

18.2 WDT Registers Address Table

Table 19-1 WDT registers

Address	Register	Name
700Ah	P_Watchdog_Ctrl	Watchdog control register
700Bh	P_Watchdog_Clr	Watchdog clear register

18.3 Control Registers

18.3.1 P_WatchDog_Ctrl (0x700A) : Watchdog Control Register

This register provides the watchdog clear timer and on/off function for firmware setting.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
WDEN	WDRS	Reserved					
B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDCHK					WDPS		

Bit15 **WDEN**: Watchdog timer enable bit. This bit enable/disable the watchdog timer. Once enabled, the watchdog timer can not be disabled by software. This bit will be reset to default value when a system reset is generated.

0 = Disable

1 = Enable

Bit14 **WDRS**: Watchdog reset select bit

This bit select whether a system or CPU reset be generated when watchdog timer overflows.

0 = System reset

1 = CPU reset

Bit 13:8 Reserved

Bit 7:3 **WDCHK**: Watchdog control register check bits. To change the settings of P_WatchDog_Ctrl register, "10101" must be written to these bits. Otherwise a watchdog reset will be generated. These bits will be read as '0'.

Bit 2:0 **WDPS**: Watchdog Timer Time-out Selections

The following table summarizes the watch-dog time-out time at FCK is 24.0 MHz.

Table 19-2 WDT Time-out

WDPS	WDT Clock Rate (Hz)	Time-out Time (FCK=24MHz)
000	FCK/65536	699.05ms
001	FCK/32768	349.52ms
010	FCK/16384	174.76ms
011	FCK/8192	87.38ms
100	FCK/4096	43.69ms
101	FCK/2048	21.84ms
110	FCK/1024	10.92ms
111	FCK/512	5.46ms

18.3.2 P_WatchDog_Clr (0x700B) : Watchdog Clear Register

P_WatchDog_Clr register is used to clear watchdog timer. Write 0xA005 to clear watchdog timer. A watchdog reset will be generated if other value has been written.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDTCLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDTCLR							

18.4 Design Tips

【Example 19-1】 WatchDog timer reset @ FCK / 32768.

```

P_WatchDog_Ctrl->W = CW_WDEN + CW_WDRS_CPU_Reset + CW_WDCHK_Setting +
                      CW_WDPS_FCKdiv32768;           /* enable WatchDog */
while(1)
    P_WatchDog_Clr->W = CW_WatchDog_Clear;           /* clear watchdog timer */

```

Listing 19-1 WatchDog timer setting

19 Instruction Set

19.1 Introduction

The following table 20-1 shows all instruction types, operation styles, and execution cycles. The contents of table means:

1. **RW**: Memory Read Waiting cycle, RW if no wait state insertion and RW = N if wait state = N.
2. **SW**: Memory Write Waiting Cycle, SW= 0 ~N. = 0 ~ N
3. **RW** : store or read waiting cycle, SRW = SW when ALU = store else SRW = RW.

Table 20-1: instruction types

Type	Operation	Cycles
JMPF	Goto label	5+2RW
DSI6	DS=I6	2+RW
JMPR	Goto MR	4+RW
CALL	CALL label	9+2RW+2SW
FIR_MOV	FIR_MOV_ON/OFF	2+RW
Fraction	Fraction ON/OFF	2+RW
INT SET	INT FIQ/IRQ	2+RW
IRQ	IRQ ON/OFF	2+RW
SECBANK	SECBANK ON/OFF	2+RW
FIQ	FIQ ON/OFF	2+RW
IRQ Nest Mode	IRQNEST ON/OFF	2+RW
BREAK	BREAK	10+2RW+2SW
CALLR	CALL MR	8+RW+2SW
DIVS	DIVS MR,R2	2+RW
DIVQ	DIVQ MR,R2	3+RW
EXP	R2= EXP R4	2+RW
NOP	NOP	2+RW
DS Access	DS=Rs/ Rs=Ds	2+RW
FR Access	FR=Rs/ Rs=FR	2+RW
MUL	MR = Rd* Rs, {ss,us,uu}	12+RW / 13+RW (uu)
MULS	MR = [Rd]*[Rs], size,{ss,us,uu}	us,ss : 10*N+6 + (N+1)*2*RW + {N*SW} / uu: 11*N+6 + (N+1)*2*RW + {N*SW}
Register BITOP	BITOP Rd,Rs	4+RW
Register BITOP	BITOP Rd,offset	4+RW
Memory BITOP	BITOP DS: [Rd],offset	7+2RW+SW
Memory BITOP	BITOP DS: [Rd],Rs	7+2RW+SW
Shift	Rd=Rd LSFT Rs	8+RW
RETI	RETI	8+3RW / 10+4RW (IRQ NEST ON)
RETF	RETF	8+3RW

Type	Operation	Cycles
Base+Disp6	Rd = Rd op [BP+IM6]	6+2RW
Imm6	Rd = Rd op IM6	2+RW
Branch	Jxx label	2+RW / 4+RW (taken)
Indirect	Push/Pop Rx,Ry to [Rs]	4+ 2N + (N+1)RW
DS_Indirect	Rd = Rd op DS: [Rs++]	6+RW+SRW / 7+RW+SRW (PC)
Imm16	Rd = Rs op IMM16	4+2RW / 5+2RW (PC)
Direct16	Rd = Rs op A16	7+2RW+SRW / 8+2RW+SRW (PC)
Direct6	Rd = Rd op A6	5+RW+SRW / 6+RW+SRW (PC)
Register	Rd = Rd op Rs SFT sfc	3+RW / 5+RW (PC)

- (a) MULS Cycle: $10*N+6 + (N+1)*2^*RW + \{N^*SW\}$ (signedxsigned, unsignedxsigned)
 $11*N+6 + (N+1)*2^*RW + \{N^*SW\}$ (unsignedxunsigned) where N=1..16, (N^*SW) = 0 if FIR_MOVE OFF
- (b) DS_Indirect Cycle: $6 + RW + SRW / 7 + RW + SRW$ (write to PC)
- (c) Direct16 Cycle: $7 + 2^*RW + SRW / 8 + 2^*RW + SRW$ (write to PC)
- (d) Direct6 Cycle: $5+ RW + SRW / 6+ RW + SRW$ (write to PC)
- (e) RW: Memory Read Waiting cycle, RW= 0 ~ N, SW: Memory Write Waiting Cycle, SW= 0 ~N.
SRW: store or read waiting cycle, SRW = SW when ALU = store else SRW = RW.
- (f) D: 0 (forward jump) / 1 (backward jump)
W: 0 (not store) / 1 (store)
DS: 0 (not using DS) / 1 (using DS)

19.2 ALU Operation

The table 20-2 shows the relationship between operation and flag. The flag have 4 components, N, Z, S, C.

N: Negative

Z: Zero

S: Sign

C: Carry

Table 20-2: relationship between operation and flag

Operation Type	Operation	N	Z	S	C	Example
Addition	a + b	✓	✓	✓	✓	Rd=Rd + Rs
Add with carry	a + b + c	✓	✓	✓	✓	Rd=Rd + Rs,c
Subtraction	a +~ b + 1	✓	✓	✓	✓	Rd=Rd - Rs
Sub with carry	a +~ b + c	✓	✓	✓	✓	Rd=Rd - Rs,c
Compare	a +~ b + 1	✓	✓	✓	✓	Cmp Rd,Rs
Negative	~ b + 1	✓	✓	-	-	Rd= ~Rs
Exclusive OR	a xor b	✓	✓	-	-	Rd=Rd ^ Rs
Load	a = b	✓	✓	-	-	Rd=Rs
OR	a or b	✓	✓	-	-	Rd=Rd Rs
AND	a and b	✓	✓	-	-	Rd=Rd & Rs
Test	Test a, b	✓	✓	-	-	Test Rd, Rs
Store	Store	-	-	-	-	[Rd]=Rs

19.3 Addressing Modes

The unSP ISA V1.2 CPU support the following six addressing modes :

Register

Users can shift the source register (Rs) value first and then executing ALU operation with destination register (Rd), place the result at destination register.

Immediate

Users can do ALU operation between source register and a 6-bits or a 16-bits immediate value, then place the result at destination register.

Direct

- a. Users can do alu operation between source register and the value at memory location indexed by 6-bits or 16-bits operand, then place the result at destination register
- b. Users can store register value to a memory location indexed by 6-bits or 16-bits operand.

Indirect

- a. Users can do alu operation between destination register and the value at memory location indexed by source register, then place the result at destination register.
- b. Users can store destination register value to a memory location indexed by source register.
- c. The source register can be increased by 1 before alu operation or increased/decreased by 1 after alu operation.
- d. Users can use the “D:” indicator to access memory location larger than 64k words, if the “D:” indicator is used, the high 6-bits of accessing address will use data segment (DS) value or be zeroed.

Multi-indirect

Users can push or pop multiple registers' value to memory location indexed by stack pointer (SP)

Displacement

Users can do alu operation between destination register and the value at memory location indexed by base pointer (BP) with a 6-bits displacement.

19.4 Register

The SPMC series provide two kind of internal register for calculation. One is R1 ~ R4, the other is SR1 ~ SR4. User can switch secondary register bank mode by secbank on/off to change two of them, when shadow register mode is on, all operation with R1-R4 will map to SR1-SR4. The register address and relation are shown in the table 20-3.

Table 20-3: two kind of internal register for calculation

Rd/Rs	SECBANK OFF	SECBANK ON
000	SP	SP
001	R1	SR1
010	R2	SR2
011	R3	SR3
100	R4	SR4
101	BP	BP
110	SR	SR
111	PC	PC

Note: Rd: Destination Register Rs: Source Register

The SPMC also has general registers: SP (Stack pointer), SR (Status register), and FR (Flag Register). The SP is FILO structure and indicates the stack depth. Normally, user must set the SP value from start address of MCU. The SR shows the status of MCU. It indicates the code or data segment and flag for "negative", "zero", "sign" and "carry". The bit relation is shown in Table 20-4.

Table 20-4: SR: (Status Register)

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
DS						N	Z	S	C	CS					

CS: Code Segment

DS: Data Segment

N: Negative Flag

Z: Zero Flag

S: Sign Flag

C: Carry Flag

The FR shows the MCU statue flag and the description is shown in Table 20-5.

Table 20-5 FR: (Flag Register)

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
-	AQ	BNK	FRA	FIR	SFT BUF						F	I	INE	IRQ PRI	

AQ: DIVS/DIVQ AQ bit flag.

BNK: Secondary Bank Registers Mode.

FRA:	FRACTION MODE.
FIR:	FIR_MOVE MODE.
SFT BUF:	Shift buffer/Guard Bit (FIR).
F:	FIQ enable flag.
I:	IRQ enable flag.
INE:	IRQ Nest MODE.
IRQ PRI:	IRQ Priority register.

19.5 Branch condition

The following table 16-6 shows the branch conditions. The condition match must refer to SR (Status register). Four types of branch flags are: N (negative), Z(zero), S(sign), and C(Carry). When user uses some operations to distinguish the program flow, the NZSC flag will be changed. And programmer can use the following instruction to make program jump for matching address. program jump instruction is shown in Table 20-6.

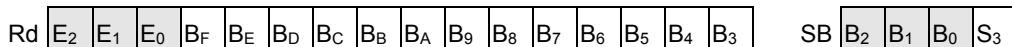
Table 20-6: program jump instruction

Syntax	Description	Branch
JCC	carry clear	C==0
JB	Below (unsigned)	C==0
JNAE	not above and equal (unsigned)	C==0
JCS	carry set	C==1
JNB	not below (unsigned)	C==1
JAE	above and equal (unsigned)	C==1
JSC	sign clear	S==0
JGE	great and equal (signed)	S==0
JNL	not less (signed)	S==0
JSS	sign set	S==1
JNGE	not great than (signed)	S==1
JL	Less (signed)	S==1
JNE	not equal	Z==0
JNZ	not zero	Z==0
JZ	zero	Z==1
JE	equal	Z==1
JPL	plus	N==0
JMI	minus	N==1
JBE	below and equal (unsigned)	Not (Z==0 and C==1)
JNA	not above (unsigned)	Not (Z==0 and C==1)
JNBE	not below and equal (unsigned)	Z==0 and C==1
JA	above (unsigned)	Z==0 and C==1
JLE	less and equal (signed)	Not (Z==0 and S==0)
JNG	not great (signed)	Not (Z==0 and S==0)

Syntax	Description	Branch
JNLE	not less and equal (signed)	Z==0 and S==0
JG	great (signed)	Z==0 and S==0
JVC	not overflow (signed)	N == S
JVS	Overflow (signed)	N != S
JMP	Unconditional branch	Always

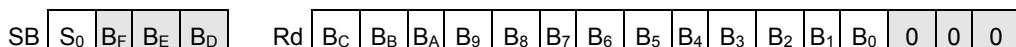
19.6 Shift

ASR: (Shift Right with MSB of Rs)

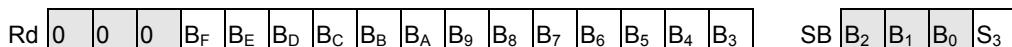


E₂E₁E₀ are signed extension bit of the most significant bit in Rs

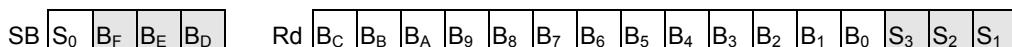
LSL: (Shift Left)



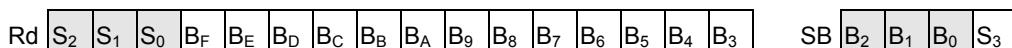
LSR: (Shift Right)



ROL: (Rotate Left with SB)



ROR: (Rotate Right with SB)



19.7 Instruction Set

N	Z	S	C
-	-	-	-

【Note】: “ - ” isn't effect to sign; “ √ ” is effect to sign;

Segmented Far Call (CALL)

N	Z	S	C
-	-	-	-

【Description】: A far call instruction with code segment assigned; both PC and SR are pushed to stack automatically.

【 Cycles 】: 9 + 2*RW + 2*SW (RW: Read Waiting Cycle . SW: Write Waiting Cycle)

【 Word 】: 2

【 Version 】: ALL Version.

【 Syntax 】: CALL Label;

【 Example 】:

```
.CODE
.PUBLIC    _main
_main:
    r1 = 0x0001
    r1 += 0x0002
    call F_Sub           //Call subroutine
    r1 += r4

L_Loop:
    nop
    jmp L_Loop

F_Sub:
    r2 = 0x5555
    r3 = 0x5555
    mr = r2*r3          //r2 multiply r3
    retf
```

Segmented Far Indirect Call (CALL MR)

N	Z	S	C
-	-	-	-

【 Description 】 : A far call instruction with MR register; both PC and SR are pushed to stack automatically.

【 Cycles 】 : 8 + RW + 2*SW

【 Word 】 : 1

【 Version 】 : ISA 1.2 and above.

【 Syntax 】 : CALL MR;

【 Example 】 : In this example the value of MR points to F_Sub.

```
.CODE
.PUBLIC _main
_main:
    r1 = 0xefff;
    r2 = 0x1713;
    r3 = OFFSET F_Sub           // the value of r3 is the "offset"
    r4 = SEG F_Sub             //the value of r4 is the "segment"
    call mr

L_MainLoop:
    nop
    jmp L_MainLoop

F_Sub:
    r2 = 0x5555
    r3 = 0x5555
    mr = r2*r3                //r2 multiply r3
    retf
```

GOTO (Far Jump)

N	Z	S	C
-	-	-	-

【Description】: Go to user's specified address. Target address was not be limited by 64K words.

【 Cycles 】: 5 + 2*RW

【 Version 】: ISA 1.1 and above.

【 Syntax 】: GOTO LABEL;

【 Example 】:

```
.CODE
.PUBLIC _main
_main:
    r1 = 0x000a
    r2 = 0x0001
    cmp r1,r2          //compare the bigger between r1 And r2
    jnae L_OK           //r1 is bigger
    goto L_Exchange_Value //Exchange r1 And r2 While r1 Smaller r2
L_OK:
    nop
    jmp L_OK
L_Exchange_Value:
    r3 = r1
    r1 = r2          //Exchange r1 And r2
    r2 = r3
    jmp L_OK
```

GOTO(Far Indirect JMP)

N	Z	S	C
-	-	-	-

【Description】: A far indirect jump instruction with MR register. The content of MR {R4[5: 0],R3[15: 0]} will be used as destination address.

【 Cycles 】: 4 + RW

【 Version 】: ISA 1.2 and above.

【 Syntax 】: GOTO MR;

【 Example 】:

```
.CODE  
.PUBLIC _main  
  
_main:  
    r4 = SEG L_GOTO_Sub      //the value of r4 is the "segment"  
    r3 = OFFSET L_GOTO_Sub    //the value of r3 is the "offset"  
    goto mr                  // the value of mr(r4: r3) is L_GOTO_Sub  
    nop  
  
L_GOTO_Sub:  
    nop  
    jmp L_GOTO_Sub
```

RETF (Return from subroutine)

N	Z	S	C
-	-	-	-

【Description】: RETF pops SR and PC from stack and return from subroutine. Note that the SR and PC are popped back after RETF. Therefore, the SR is the same as at the time when Call is made.

【 Cycles 】: 8 + RW + 2*SW

【 Word 】: 1

【 Version 】: All Version.

【 Syntax 】: RETF;

【 Example 】:

```
.CODE
.PUBLIC _main
_main:
    nop
    call F_Delay1
_L_loop:
    nop
    jmp L_loop

F_Delay1:
    nop
    retf
```

RETI (Return from interrupt)

N	Z	S	C
-	-	-	-

【Description】: if IRQ nest mode on RETI pops FR, SR, PC from stack else RETI pops SR and PC from stack then return from interrupt service routine and clear internal FIQ or IRQ interrupt flag. Note that the SR is pop back after RETI. Therefore, the SR is the same as at the time when IRQ is executed. Also, a post-FIQ is unable to interrupt existed FIQ.

【 Cycles 】: 8 + 3*RW (IRQ Nest Mode OFF) / 10 + 4*RW (IRQ Nest Mode ON)

【 Word 】: 1

【 Version 】: All Version.

【 Syntax 】: RETI;

Break (Software Interrupt)

N	Z	S	C
-	-	-	-

【Description】: Generate a software interrupt. System will go to [0x0x00ffff].

【 Cycles 】: 10 + 2*RW + 2*SW

【 Word 】: 1

【 Version 】: ALL Version.

【 Syntax 】: BREAK;

【 Example】:

```
.CODE
.PUBLIC _main
_main:
    r1 = 0x1234
    BREAK          // generate software interrupt
    L_MainLoop:
        nop
        jmp    L_MainLoop

.TEXT
.PUBLIC _BREAK
_BREAK:
    push  r1, r5  to  [sp]
    nop           //Write Break Function In this Area
    pop   r1, r5  from  [sp]
    reti          //return to the main function
```

FIR_MOV ON/OFF

N	Z	S	C
-	-	-	-

【Description】 : Enable/Disable automatic data movement for FIR operations. It affects the behavior of FIR. The status is a global behavior. Hence, use it in interrupt with care.

【 Cycles 】 : 2 + RW

【 Word 】 : 1

【 Version 】 : ALL Version.

【 Syntax 】 : FIR_MOV ON/OFF;

【 Example 】 :

```
.IRAM
    .VAR    NO_1=0x0001, NO_2=0x0002,      NO_3=0x0003, NO_4=0x0004
    .VAR    NO_5=0x0005, NO_6=0x0006,      NO_7,          NO_8

.CODE
.PUBLIC _main
_main:
    FIR_MOV ON           //Enable automatic data movement for FIR operations.
    r1 = NO_2             //Give the address of "NO_2" to 'r1'
    r2 = NO_5             //Give the address of "NO_5" to 'r2'
Loop_Muls:
    mr = [r1]*[r2],us,2   //The result of operation is stored in r4: r3
    [NO_7] =r3            //Read the result (low 16 bite) to NO_7
    [NO_8] =r4            //Read the result (high 16 bite) to NO_8
    nop                  //The value of r1,r2 both added 2
    jmp     Loop_Muls
```

Secondary Register Bank Mode ON/OFF

N	Z	S	C
-	-	-	-

【Description】: Switch secondary register bank mode ON/OFF, 4 shadow registers SR1-SR4 are added in μ'nSP™ ISA Version 1.2 , user can use this instruction to switch secondary register bank mode on/off, when shadow register mode is on, all operation with R1-R4 will map to SR1-SR4.

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】: SECBANK ON/OFF;

【 Example 】: .CODE

```
.PUBLIC _main
_main:
    SECBANK ON
    r1 = 0x1234          // 0x1234 is sent to SR1 , no value in r1
    r2 = 0x5678          // 0x5678 is sent to SR2 , no value in r2
    r3 = 0x0f0f          // 0x0f0f is sent to SR3 , no value in r3
    r4 = 0xf0f0          // 0xf0f0 is sent to SR4 , no value in r4
_L_Loop:
    nop
    jmp _L_Loop
```

Fraction Mode ON/OFF

N	Z	S	C
-	-	-	-

【Description】: Switch Fraction Mode ON/OFF, when Fraction mode is ON, the 32 bit result of multiplication will be shift 1 bit left.

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】: FRACTION ON/OFF;

【 Example 】:

```
.CODE
.PUBLIC _main
_main:
    r1 = 0x0002
    r2 = 0x2244
    FRACTION ON
    mr = r1*r2      //Case1: "FRACTION OFF" the result(R4R3) is 0x00004488
                  //Case2: "FRACTION ON" the result(R4R3) is 0x00008910
_L_Loop:
    nop
    jmp _L_Loop
```

IRQ Nested Mode ON/OFF

N	Z	S	C
-	-	-	-

【Description】: Switch IRQ Nest Mode ON/OFF, when IRQ Nest Mode is ON, CPU will push FR/SR/PC into stack before enter IRQ routine and pop them out when leaving, and higher priority IRQ interrupting is allowed when executing IRQ routine, user can set-up IRQ PRI register to change the interrupt priority

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】: IRQNEST ON/OFF;

IRQ ON/OFF

N	Z	S	C
-	-	-	-

【Description】: Enable/Disable IRQ

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ALL Version.

【 Syntax 】: IRQ ON/OFF; //Open or close IRQ interrupt

【 Example 】:

```
.include hardware.inc
.RAM
.VAR R_Value
.CODE
.PUBLIC _main
_main:
    r1 = 0xffff          //Set PORTA as output
    [P_IOA_Dir] = r1
    [P_IOA_Attrib] = r1
    [P_IOA_Data] = r1
    r1 = 0x0001          //Set 2Hz Inturrupt
    [P_INT_Ctrl] = r1
    IRQ ON              //Set IRQ ON

L_Loop:
    nop
    jmp L_Loop

.TEXT
.PUBLIC _IRQ7
_IRQ7:
    push r1,r5 to [SP]
    r4 = [R_Value]
    r4 ^= 0xffff
    [R_Value] = r4
    [P_IOA_Data] = r4
    r1 = 0x0001          //Clear 2Hz interrupt flag
    [P_INT_Clear] = r1
    pop r1 ,r5 from [SP]
    reti
```

FIQ ON/OFF

N	Z	S	C
-	-	-	-

【Description】: Enable/Disable FIQ

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ALL Version.

【 Syntax 】: FIQ ON/OFF; //Open or close FIQ interrupt

Interrupt Set

N	Z	S	C
-	-	-	-

【Description】: Set FIQ/IRQ flags.

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version】: All Version.

【 Syntax 】:

```
INT    FIQ;                      //enable FIQ, (disable IRQ)
INT    IRQ;                      //enable IRQ, (disable FIQ)
INT    FIQ, IRQ;                 //enable FIQ and IRQ
INT    OFF;                      //disable FIQ and IRQ
```

【 Example 】:

```
INT    FIQ;                      // enable FIQ, (disable IRQ)
INT    FIQ, IRQ;                 // enable IRQ, FIQ
```

DS Segment Direct Access Instruction

N	Z	S	C
-	-	-	-

【Description】: Set and get DS segment register instruction, Since the DS-Segment is a 6 bit register, only the LSB 6 bit of Rs will be put on DS when executing set DS operation and zero-extended is used when executing get DS operation.

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】:

```
DS = RS;  
RS = DS;
```

【 Example 】:

```
.CODE  
.PUBLIC _main  
_main:  
    r1 = 0xff1f      //Send LSB6 bite of r1 to DS  
    DS = r1          //From the register of "SR" we can find it  
L_Loop :  
    nop  

```

Change DS Segment with immediate 6-bit value

N	Z	S	C
-	-	-	-

【Description】: Set DS segment register with 6-bit immediate value instruction.

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】: DS = I6;

【 Example 】:

```
.CODE  
.PUBLIC _main  
  
_main:  
    DS = 0xfffff //From the register of "SR" we can find it's change
```

L_Loop :

```
    nop  
    jmp L_Loop
```

NOP

N	Z	S	C
-	-	-	-

【Description】: No operation, only increase PC to next address.

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】: NOP;

Processor Flag Access Instruction*

N	Z	S	C
-	-	-	-

【Description】 : Direct access the Processor Flag Register, the content of register is show below.

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
-	AQ	BNK	FRA	FIR		SFT BUF			F	I	INE		IRQ PRI		

AQ: DIVS/DIVQ AQ bit flag, default is 0.

BNK: Secondary Bank Registers Mode, default is 0.

FRA: FRACTION MODE, default is 0.

FIR: FIR_MOVE MODE, default is 0. (FIR_MOVE ON : 0 FIR_MOVE OFF : 1)

SFT BUF: Shift buffer/Guard Bit (FIR), default is 4'b0000.

F: FIQ flag, default is 0.

I: IRQ flag, default is 0.

INE: IRQ Nest MODE, default is 0.

IRQ PRI: IRQ Priority register, default is 1000 after reset, if any IRQ occurred, IRQ PRI register will be set as the IRQ priority, only the IRQ with higher priority can interrupt it, user can customize the IRQ Nesting with setting priority register.

Priority: IRQ0>IRQ1>IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7

IRQ Enable algorithm : 0 – PRI -1

Note: FIQ still has highest priority than any IRQ if FIQ is enable.

For example:

- 1、 if PRI is 1000, all IRQ 0-7 are enable
- 2、 if PRI is 0000, all IRQ 0-7 are disable
- 3、 IRQ3 occurred, PRI will be set to 0011, only IRQ0-2 are enable

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】:

FR = Rs ;

Rs = FR ;

DIVS/DIVQ

N	Z	S	C
-	-	-	-

【Description】: These instructions implement 32-bit division. There are two divide primitives, DIVS and DIVQ. A single precision divide, with a 32-bit numerator and a 16-bit denominator, yielding a 16-bit quotient, executes in 16*3 cycles. Higher precision divides are also possible. The division can be either signed or unsigned, but both the numerator and denominator must be the same. Place the 32-bit numerator at R4: R3, the 16-bit denominator at R2 and clear the AQ flag then executed with the divide primitives, DIVS and DIVQ. Repeated execution of DIVQ implements a non-restoring conditional add-subtract division algorithm. At the conclusion of divide operation the quotient will be placed at R3.

To implement a signed divide, first execute the DIVS instruction once, which computes the sign of the quotient. Then execute the DIVQ instruction for as many times as there are bits remaining in the quotient.

【 Cycles 】: 2 + RW (DIVS) / 3 + RW (DIVQ)

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】:

DIVS MR,R2;

DIVQ MR,R2;

【 Example1 】: Divide a 32 bit signed number(0xfffff1713) with 16-bit divisor(divide 0x0625)

```
.CODE
.PUBLIC _main
_main:
    r4 = 0xffff;
    r3 = 0x1713;
    r2 = 0x0625;
    r1 = 0x0001
    r4 = r4 lsl r1;           //shift the dividend 1 bit left for (31.1) format
    mr |= r3 lsl r1         //change to follows
    r1 = fr;
    clrb r1,0xe;            //clear AQ flag
    fr = r1;
    r1 = 0;
    DIVS mr,R2;             //divide the signed bit
```

L_Signed:

```
DIVQ mr,R2;           //divide the remainder
r1 += 1;
cmp r1,0xf;
jne L_Signed;
r1 = r3;             //the quotient is in R3,that is 0xffda

L_Loop:
nop;
jmp L_Loop
```

【 Example2 】 : Divide a 32 bit unsigned number(0x00031713) with 16-bit divisor(0x0625)

```
.CODE
.PUBLIC _main
_main:
r4 = 0x0003;
r3 = 0x1713;
r2 = 0x0625;
r1 = 0x0001
r4 = r4 lsl r1;      // shift the dividend 1 bit left for (31.1) format
mr |= r3 lsl r1     // change to follows
r1 = fr;
clrb r1,0xe;         // clear AQ flag
fr = r1;
r1 = 0;
L_unsigned:
DIVQ mr,R2;          // divide the remainder
r1 += 1;
cmp r1,0x10;
jne L_unsigned;
r1 = r3;             //the quotient is in R3,that is 0x0080

L_Loop:
nop;
jmp L_Loop
```

EXP

N	Z	S	C
-	-	-	-

【Description】: The EXP instruction derives the effective exponent of the R4 register to prepare for the normalization operation, and places the result in the R2. The result is equal to the number of redundant unsign(sign) bits in the input. The truth table is as follows:

Rsrc	Rdst	Result
SNDDDDDD DDDDDDDD	0	
SSNDDDDD DDDDDDDD	1	
SSSNDDDD DDDDDDDD	2	
SSSSNDDD DDDDDDDD	3	
SSSSSNDD DDDDDDDD	4	
SSSSSSND DDDDDDDD	5	
SSSSSSSN DDDDDDDD	6	
SSSSSSSS NDDDDDDD	7	
SSSSSSSS SNDDDDDD	8	
SSSSSSSS SSNDDDDD	9	
SSSSSSSS SSSNDDDD	10	
SSSSSSSS SSSSNDDD	11	
SSSSSSSS SSSSSNDD	12	
SSSSSSSS SSSSSND	13	
SSSSSSSS SSSSSSSN	14	
SSSSSSSS SSSSSSSS	15	

Note:

S = Sign Bit

N = Non-Sign Bit

D = Don't care Bit

【 Cycles 】: 2 + RW

【 Word 】: 1

【 Version】: ISA 1.2 and above.

【 Syntax 】: R2= EXP R4;

【Example】:

```
.CODE
.PUBLIC _main
_main:
    r4 = 0x1000;      //the 15th bite of "r4" is "0"
    r2 = EXP     r4;      //the result is calculate by this way from 15th bit to the bit, which is
    opposite
                                //to 15th and sub 1, the result is 2
_L_Loop:
    nop
    jmp    L_Loop
```

Bit Operation

N	Z	S	C
-	√	-	-

【Description】: Bit_op for Rs or Memory DS: Rs (0x000000-0x3fffff), the origin value before operation will affect the zero flag, that is if the origin bit is 1 then zero flag will be false else zero flag will be true. With registers bit operation, only the least significant 4 bits of register will be used, any other bits will be ignored.

【 Bit_op 】: tstb / setb / clrb / invb

【 Cycles 】: 4 + RW (BITOP with Registers) / 7 + 2*RW + SW (BITOP with memory)

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】:

1. Memory Bit Operation

BITOP {DS: }[Rd], offset; //offset=0-15

BITOP {DS: }[Rd], Rs; //Rs=R0-R7 Rd=R0-R6

2. Register Bit Operation

BITOP Rd, offset; //offset=0-15

BITOP Rd, Rs; //Rs=R0-R7 Rd=R0-R6

BITOP :

00	01	10	11
tstb	setb	clrb	invb

【 Example1】:

```
tstb D: [r2], 13;
setb      [r1], r3;
clr b r3, 10;
invb r4, r5;
```

【 Example2】:

```
.CODE
.PUBLIC _main
_main:
    r1 = 0x0f0f
    r5 = 0x0001
    setb r1, 15      //turn "r1" to 0x8f0f
    clrb r1, 15      //turn "r1" to 0x0f0f
    invb r1, r5      //the R5 =1,so invert the 1th of r1, the result is 0x0f0d
    tstb [r1], 8      //only change the "Z"
```

L_Loop:

```
nop  
jmp L_Loop
```

Shift Operation

N	Z	S	C
√	√	-	-

【Description】: Shift_op for Rd and Rs. A 16-bit multi-cycle shifter with one instruction, and support 32-bit shifter with combining 2 shift instruction, when using 32-bit shifter the result must place at R4: R3, the rotate ROR/ROL operation will shift with carry and the drop bit will place at carry flag after operation.

【 Shift_op 】: supported ASR/ASROR/LSL/LSLOR/LSR/LSROR/ROL/ROR

【 Cycles 】: 8 + RW

【 Word 】: 1

【 Version 】: ISA 1.2 and above.

【 Syntax 】:

Rd = Rd shift_op Rs; Rd: destination register (only R0-R6 is available)

Rs: shift count register

Rs[4: 0] valid : ASR, ASROR, LSL, LSLOR, LSR, LSROR

Rs[3: 0] valid: ROL, ROR

Shift_OP :

000	001	010	011	100	101	110	111
ASR	ASR-OR	LSL	LSL-OR	LSR	LSR-OR	ROL	ROR

【 Example1 】:

```
r2 = r2 asr r1;           // 16-bit arithmetic right shift
r3 = r3 lsr r1;           // 32-bit arithmetic right shift
mr |= r4 asr r1           // result will be put r4: r3
```

【 Example2 】:

```
r1 = 0x1234
r1 = r1 rol 4             // the value is 0x2340
r1 = r1 rol 4             // the value is 0x3401
r1 = r1 rol 4             // the value is 0x4012
r1 = r1 rol 4             // the value is 0x0123
r1 = r1 rol 4             // the value is 0x1234

r1 = 0xffff
r1 = r1 lsr 4             // the value is 0x0fff
r1 = r1 rol 4             // the value is 0xffff
r1 = r1 lsl 4             // the value is 0xffff0
r1 = r1 ror 4             // the value is 0xffff
r1 |= r1 lsr 4            //First r1 lsr 4 bite then OR the origin"r1"

r1 = 0xff00                //the origin value of "r1" change to 0xff00
r2 = 0x000                 //the origin value of "r2" is 0x0007
```

```
r1 = r1 asr r2           //the result is 0xffffe
```

【 Example3 】:

```
r1 = 0x0006
r3 = 0x0f0f
r4 = 0xf0f0
r3 = r3 lsr r1           //first r3 lsr 6 bite
mr |= r4 asr r1          //put some bit of r4 to r3; r3=0xc03c,r4=0ffc3
```

Registers Multiplication (Mul)

N	Z	S	C
-	-	-	-

【Description】: Multiply two registers (signed to signed or unsigned to signed) and place result in registers R4: R3 (MR). If Fraction mode is enable the result of multiplication will be shift 1-bit left. The Rd register only support R0-R6

【 Cycles 】: 12+RW (signedxsigned, unsignedxsigned) / 13+RW (unsignedxunsigned)

【 Word 】: 1

【 Version 】: (signedxsigned, unsignedxsigned) All Version / (unsignedxunsigned) ISA 1.2 and above.

【 Syntax 】:

```
MR = Rd * Rs;           // signed-to-signed multiplication used if no indication  
MR = Rd * Rs,uu;        // unsigned-to-unsigned  
MR = Rd * Rs, ss;       // signed-to-signed  
MR = Rd * Rs, us;       // Rd is unsigned and Rs is singed. The first register is always  
                        unsigned  
                        // and the second register is signed.
```

【 Example1】:

```
r1 = 0x8002  
r2 = 0x0002           // the result is stored in R4: R3  
mr = r1*r2            // 12 cycle, r4=0xffff,r3=0x0004  
mr = r1*r2,ss          // 12 cycle, r4=0xffff,r3=0x0004  
mr = r2*r1,us          // 12 cycle, r4=0xffff,r3=0x0004  
mr =r1*r2,uu            // 14 cycle, r4=0x0001,r3=0x0004
```

Sum of Registers Multiplication (Muls)

N	Z	S	C
-	-	✓	-

【Description】: Use a 36-bit Arithmetic Unit to sum up a consecutive register multiplication (FIR) and propagate coefficients for next FIR. If Fraction mode is enable, the result of every multiplication will be shift 1-bit left and then sums up. The Rd and Rs pointers are adjusted automatically, if FIR_MOVE flag is On and when n >1, the contents of memory pointed by Rd are also moved forward. After operation, the 4-bit MSB (guard bit) of ALU will be placed at shift-buffer and sign flag will be set if any overflow occurred of the result.

【 Cycles 】: $10*N+6 + (N+1)*2^*RW + \{N^*SW\}$ (signedxsigned, unsignedxsigned)
 $11*N+6 + (N+1)*2^*RW + \{N^*SW\}$ (unsignedxunsigned)
where N=1..16, (N^*SW) = 0 if FIR_MOVE OFF

【 Word 】: 1

【 Version 】: (signedxsigned, unsignedxsigned) All Version /(unsignedxunsigned) ISA 1.2 and above.

【 Syntax 】:

MR = [Rd] * [Rs]{, ss}{, n};

MR = [Rd] * [Rs], us{, n};

*Signed-to-signed multiplication used if no indication; MR is cleared before summation.

*n: [16: 1]. if n is omitted, 1 is used.

n=1 Before	After
R d	R d

mem X1 	X1
R s	R s

mem C1 	C1
--	--

MR= X1*C1

n=4 Before	After
R d	R d

mem X1 X2 X3 X4 	X1 X1 X2 X3
---	---

*Note: Rd has been shifted one position to the right.

R s	R s
Mem C1 C2 C3 C4 	C1 C2 C3 C4

MR= C1*X1+C2*X2+C3*X3+C4*X4

Note: The FIR operation of previous version u'nSP (ISA 1.0 / ISA 1.1) does not change the sign flag, but the new version of u'nSP (ISA 1.2) will change this flag to indicate overflow occurrence.

【Limitation】:

Result is incorrect if the following conditions are both met:

n >1 and (either Rs or Rd are set to R3 or R4) and (Rs and Rd are set to the same register.)

【 Example】:

MR = [R1] * [R2], 8;

MR = [R1] * [R2], us, 2;

Note : Here you can refer the example in the FIR_MOV ON/OFF instruction

Conditional branch (Branch)

N	Z	S	C
-	-	-	-

【 Description 】 : A short jump instruction to local label (address within ± 63 words), and one non-conditional included. See Conditional Branch Table (Table 4) for details.

【 Cycles 】 : 2+RW (not-taken) / 4+RW (taken)

【 Word 】 : 1

【 Version 】 : All Version.

【 Syntax 】 :

```
Conditional_jump label;  
jmp label;
```

【 Example 】 :

```
jmp      LABEL;          // non-conditional jump  
jne      LABEL;          // jump when not equal
```

Push/Pop operations (By Indirect Addressing)

N	Z	S	C
-	-	-	-

【Description】 : Push/Pop a set of registers to/from memory, which is indicated by Rs consecutively.

That is also used for RETF/RETI.

【 Cycles 】 : $4 + 2 \times N + (N+1) \times RW$ // where N=1..7, the number of pop or push.

【 Word 】 : 1

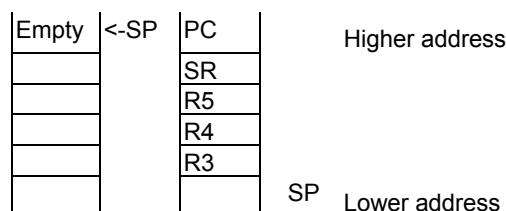
【 Version 】 : All Version.

【 Syntax 】 :

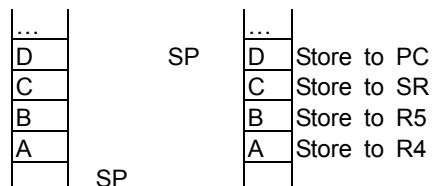
```
push Rx, Ry to [Rs] ;      //push Rx through Ry to [Rs]
pop Rx, Ry from [Rs] ;     //pop Rx through Ry from [Rs]
```

【 Example 】 :

```
Push R3, PC to [SP];      //Push R3 through PC (R7) to [SP]
```



```
pop R4, PC from [SP];      //pop R4 through PC (R7) from [SP]
pop SR, PC from [SP];      // retf
```



Note: push R1, R5 to [SP] is equivalent to push R5, R1 to [SP]

ALU operations with base plus displacement memory access (Base+Disp6)

N	Z	S	C
√	√	√	√

【Description】: Alu_op for Rd and memory [BP + IM6], or store Rd to [BP + IM6]. BP is R5.

【 ALU 】: Supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg/store

【 Cycles 】: 6+ 2*RW

【 Word 】: 1

【 Version 】: All Version.

【 Syntax 】:

```
Rd alu_op = [BP + IM6];
[BP + IM6] = Rd;
```

Note: List of all possible alu_ops:

Rd -= [BP + IM6];	// sub
Rd -= [BP + IM6], carry;	// sbc
Rd += [BP + IM6], carry;	// adc
Rd += [BP + IM6];	// add
cmp Rd , [BP + IM6];	// cmp
Rd &= [BP + IM6];	// and
Rd = [BP + IM6];	// or
Rd ^= [BP + IM6];	// xor
Rd = [BP + IM6];	// load
test Rd , [BP + IM6];	// test
Rd =- [BP + IM6];	// negative
[BP + IM6] = Rd;	// store

【 Example1】:

```
r2 += [bp + 0x20], Carry;           // r2 add memory and carry
[bp + 0x06] = r1;                  // store
```

【 Example2】:

```
.IRAM
T_Buffer: .DW 0x1111, 0x1111, 0x1111, 0x1111;

.....                                //The origin value of bp is 0
bp = T_Buffer;
r1 = [bp+1]                          //Get the value of memory address [bp+1] and store to r1
.....
```

ALU operations with 6-bit immediate (Imm6)

N	Z	S	C
√	√	√	√

【Description】: Alu_op for Rd and IM6

【 ALU 】: supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg

【 Cycles 】: 2+RW

【 Word 】: 1

【 Version 】: All Version.

【 Syntax 】:

Rd ALU_OP = IM6;

Note: List of all possible alu_ops:

```

Rd := IM6;                         // Sub
Rd := IM6 , Carry;                // Sbc
Rd += IM6 , Carry;                // Adc
Rd += IM6;                        // Add
cmp Rd , IM6;                    // Cmp
Rd &= IM6;                        // And
Rd |= IM6;                        // Or
Rd ^= IM6;                        // Xor
Rd = IM6;                        // Load
test Rd , IM6;                    // Test
Rd == IM6;                        // Negative

```

【 Example1】:

r1 += 0x20, Carry; //r1=r1+0x20+carry

【 Example2】:

```

r1 = 0x00ef                      // 3 cycles the origin value of r1 is 0x00ef
r2 = 0x0073
r3 = 0xffff
r1 -= 0x0001                    // the result turns to 0x00ee
r2 += 0x0030                    // the result turns to 0x00a3
r3 &= 0x0002                    // the result turns to 0x0002

```

【Examples3】:

```

r1 = 0x0002                      // 3 cycles
cmp r1,0x0010                  // 3 cycles compare r1 with 0x0010
jbe L_Lower                     // if smaller jump to the other place
nop                              // 3 cycles, go down
.....

```

L_Lower:

nop

nop

.....

ALU operations with indirect memory access and optional increment/decrement (DS_Indirect)

N	Z	S	C
√	√	√	√

【 Description 】 : Alu-op with indirect memory access and optional Increment/Decrement. This instruction is the only instruction that can access both page 0 and non-zero pages.

【 ALU 】 : supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg/store

【 Cycles 】 : 6 + RW + SRW / 7 + RW + SRW (write to pc)

SRW : store or read waiting cycle, SRW = SW when ALU = store else SRW = RW.

【 Word 】 : 1

【 Version 】 : All Version.

【 Syntax 】 :

```
Rd alu_op = {D: }[Rs];  
Rd alu_op = {D: }[Rs--];  
Rd alu_op = {D: }[Rs++];  
Rd alu_op = {D: }[++Rs];
```

【 Example 】 :

```
r1 &= D: [r2++];           // r1= r1 AND (The content in the address of DS and r2),r2=r2+1  
                           // when r2 is 0xffff doing r2= r2 + 1 operation , make r2 be 0 and DS +1  
cmp  r1 , [++r2];         //Compare r1 with the content in the address of (r2+1)  
D: [r2--] =r1;           //Assignr1 to the address of DS and r2, r2=r2-1  
                           // when r2 is 0, doing r2 – 1 operation , make r2 be ffff and DS -1
```

ALU operations with 16-bit immediate (Imm16)

N	Z	S	C
√	√	√	√

【Description】: Alu_op for Rd and Rs register.

【 ALU 】: supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg

【 Cycles 】: 4+2*RW / 5+2*RW (write to PC)

【 Word 】: 2

【 Version 】: All Version.

【 Syntax 】:

Rd = Rs alu_op IM16;

【 Example2】:

```
.CODE
.PUBLIC _main
_main:
    r1 = 0xffff
    r2 = 0
    L_Imm16:
        cmp r1,0x00ff          //compare r1 with 16 immediate 0x00ff
        jb  L_Loop
        r1 -= 0x00ff           //sub 0x00ff
        r2 += 1
        goto L_Imm16
    L_Loop:
        jmp   L_Loop
```

ALU operations with direct memory addressing (Direct 16)

N	Z	S	C
√	√	√	√

【Description】: Alu_op for Rd, Rs and Memory A16. Two distinct modes applied by setting W bit.

【 ALU 】: supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg

【 Cycles 】: 7 + 2*RW + SRW / 8 + 2*RW + SRW (write to PC)

【 Word 】: 2

【 Version 】: All Version.

【 Syntax 】:

Rd = Rs alu_op [A16];

[A16] = Rs alu_op Rd;

【 Example1】:

bp = r1 + [Q_table]; // bp = r1 + content of Q_table where Q_table is in 0x0000 ~ 0xffff

[Hashing_tbl] = -bp; // Assign -bp to Hashing_tbl

[Q_Table+10] = bp ^ r2; // The result of bp xor r2 is assigned to Q_Table+10

r1 = [0x8079]; //Get the value of address 0x8079 and send it to r1

ALU Operations with Direct Memory Addressing (Direct 6)

N	Z	S	C
√	√	√	√

【Description】: Alu_op for Rd, and Memory A6 (0x00 ~ 0x03f).

【 ALU 】: supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg

【 Cycles 】: 5+ RW +SRW / 6+ RW +SRW (write to PC)

【 Word 】: 1

【 Version 】: All Version.

【 Syntax 】:

Rd = Rd alu_op [A6]

Rd = [A6]

[A6] = Rd

【 Example1 】:

```
.RAM
.VAR Init_Q;           // Init_Q is an variable located in memory 0x00~0x3f
.CODE
.public _main
_main:
    r1= [0x0009]      //get the value of address 0x0009and send it to r1
    r2 = 0x5555;
    bp = [Init_Q];    // load content of Init_0 to bp
    [Init_Q] = R2;    // load r2 to Init_0
    ....
```

ALU Operations with Shift (Register)

N	Z	S	C
√	√	√	√

【Description】: Alu_op for Rd and Rs. Additional shift operation can be applied to Rs before alu_op.

【 ALU 】: supported sub/sbc/adc/add/cmp/and/or/xor/load/test/neg/store

【 Cycles 】: 3 + RW / 5 + RW (write to PC)

【 Word 】: 1

【 Version 】: All Version.

【 Syntax 】:

Rd {alu_op} = Rs {}, Carry}

Rd {alu_op} = Rs ASR n {}, Carry}

Rd {alu_op} = Rs LSL n {}, Carry}

Rd {alu_op} = Rs LSR n {}, Carry}

Rd {alu_op} = Rs ROL n {}, Carry}

Rd {alu_op} = Rs ROR n {}, Carry}

where n is number of position shift and ranged in [4: 1]

Input:

Rs	B _F	B _E	B _D	B _C	B _B	B _A	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀	SB	S ₃	S ₂	S ₁	S ₀
----	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----	----------------	----------------	----------------	----------------

Suppose n=3, after shift op of

ASR: (Shift Right with MSB of Rs)

Rd	E ₂	E ₁	E ₀	B _F	B _E	B _D	B _C	B _B	B _A	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	SB	B ₂	B ₁	B ₀	S ₃
----	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----	----------------	----------------	----------------	----------------

E₂E₁E₀ are signed extension bit of the most significant bit in Rs

LSL: (Shift Left)

SB	S ₀	B _F	B _E	B _D	Rd	B _C	B _B	B _A	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀	0	0	0
----	----------------	----------------	----------------	----------------	----	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	---	---	---

LSR: (Shift Right)

Rd	0	0	0	B _F	B _E	B _D	B _C	B _B	B _A	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	SB	B ₂	B ₁	B ₀	S ₃
----	---	---	---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----	----------------	----------------	----------------	----------------

ROL: (Rotate Left with SB)

SB	S ₀	B _F	B _E	B _D	Rd	B _C	B _B	B _A	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀	S ₃	S ₂	S ₁
----	----------------	----------------	----------------	----------------	----	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

ROR: (Rotate Right with SB)

Rd	S ₂	S ₁	S ₀	B _F	B _E	B _D	B _C	B _B	B _A	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	SB	B ₂	B ₁	B ₀	S ₃
----	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----	----------------	----------------	----------------	----------------

Note :

2. FIQ, IRQ and user routine has their own Shift buffers. User does not need to save shift buffer in interrupt routines.
3. high buffer values are unknown after multiplication or filter operations. User should make no assumption to its value after the operations.

【 Example】:

```
bp += r1 asr 4, Carry;           // bp = bp + (r1/(2**4)) with Carry  
bp = r1 lsl 2;                 // bp = r1 << 2
```

20 HARDWARE SUMMARY

20.1 Programmable Registers of CPU on SPMC75X Family MCU

P_Reset_Status(R/W) (0x7006): This register shows the flag of reset status for firmware checking.

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FCHK							

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved	IIRF	IARF	Reserved	LVRF	WDRF	PORF	EXTRF

P_Clk_Ctrl (0x7007): System Clock control register

B15	B14	B13	B2	B11	B10	b9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
OSCSF	OSCIE	Reserved					

B7	B6	B5	B4	B3	Bb2	b1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

20.2 Programmable Registers of Watchdog Unit on SPMC75X Family MCU

P_WatchDog_Ctrl (R/W) (0x700A) : Watchdog Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
WDEN	WDRS	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDCHK				WDPS			

P_WatchDog_Clr (W) (0x700B) : Watchdog Clear Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	1	0	1	1	0	1	0
WDTCLR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	1	0	1	1	0	1	0
WDTCLR							

20.3 Programmable Registers of Power-Saving Unit on SPMC75X Family MCU

P_Wait_Enter (0x700C) : Wait-mode Entrance Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
WaitCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
WaitCMD							

P_Standby_Enter (0x700E) : Standby-mode Entrance Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
StandbyCMD							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	R/W
0	0	0	0	0	0	0	0
StandbyCMD							

20.4 Programmable Registers of I/O Port on SPMC75X Family MCU

P_IOA_Data (0x7060) : IO Port A Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Data							

P_IOA_Buffer (0x7061) : IO Port A Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Buffer							

P_IOA_Dir (0x7062) : IO Port A Direction Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Dir							

P_IOA_Attrib (0x7063) : IO Port A Attribute Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOA_Attrib							

P_IOA_Latch (0x7064) : IO Port A Latch Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
P_IOA_Latch							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

P_IOA_SPE (0x7080) : IO Port A Special Function Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R/W	R/W	R/W	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
Reserved	TCLKDEN	TCLKCEN	TCLKBEN	TCLKAEN	TIO2BEN	TIO2AEN	Reserved

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

P_IOA_KCER (0x7084) : IO Port A Key Change Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KC15EN	KC14EN	KC13EN	KC12EN	KC11EN	KC10EN	KC9EN	KC8WE

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

P_IOB_Data (0x7068) : IO Port B Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Data							

P_IOB_Buffer (0x7069) : IO Port B Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Buffer							

P_IOB_Dir (0x706A) : IO Port B Direction Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Dir							

P_IOB_Attrib (0x706B) : IO Port B Attribute Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOB_Attrib							

P_IOB_SPE (0x7081) : IO Port B Special Function Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved							TIO0AEN
Reserved							TIO0BEN
Reserved							TIO0CEN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	1	1	1	1	1	1
OL1EN	FTIN1EN	U1EN	V1EN	W1EN	U1NEN	V1NEN	W1NEN

P_IOC_Data (0x7070) : IO Port C Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Data							

P_IOC_Buffer (0x7071) : IO Port C Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Buffer							

P_IOC_Dir (0x7072) : IO Port C Direction Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Dir							

P_IOC_Attrib (0x7073) : IO Port C Attribute Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOC_Attrib							

P_IOC_SPE (0x7082) : IO Port C Special Function Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	0	0
W2NEN	V2NEN	U2NEN	W2EN	V2EN	U2EN	FTIN2EN	OL2EN

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TIO1CEN	TIO1BEN	TIO1AEN	Reserved	EXINT1EN	EXINT0EN	Reserved	

P_IOD_Data (0x7078) : IO Port D Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Data							

P_IOD_Buffer (0x7079) : IO Port D Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Buffer							

P_IOD_Dir (0x707A) : IO Port D Direction Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Dir							

P_IOD_Attrib (0x707B) : IO Port D Attribute Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Attrib							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
P_IOD_Attrib							

20.5 Programmable Registers of Interrupt Unit on SPMC75X Family MCU

P_INT_Status (0x70A0): Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
KEYIF	UARTIF	SPIIF	EXT1IF	EXT0IF	ADCIF	MCP4IF	MCP3IF

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TPM2IF	PDC1IF	PDC0IF	CMTIF	Reserved	OLIF	OSCSF	FTIF

P_INT_Priority (0x70A4): IRQ and FIQ Priority Selection Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
KEYIP	UARTIP	SPIIP	Reserved	EXTIP	ADCIP	MCP4IP	MCP3IP

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TPM2IP	PDC1IP	PDC0IP	CMTIP	Reserved	OLIP	OSCIP	FTIP

P_MisINT_Ctrl (0x70A8): Miscellaneous Interrupt Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
KEYIE	EXT1MS	EXT0MS	EXT1IE	EXT0IE	Reserved		

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

20.6 Programmable Registers of Timers Unit on SPMC75X Family MCU

P_TMR0_Ctrl (0x7400): Timer 0 Control Register

P_TMR1_Ctrl (0x7401): Timer 0 Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK		MODE					

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS		CKEGS					

P_TMR2_Ctrl (0x7402): Timer 2 Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPCK		MODE					

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS		CKEGS					

P_TMR3_Ctrl (0x7403): Timer 3 Control Register

P_TMR4_Ctrl (0x7404): Timer 3 Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
PRDINT		MODE					

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CCLS		CKEGS					

P_TMR_LDOK (0x740A) : Partial load prevention

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TLDCHK						LDOKE1	LDOKE0

P_TMR0_TCNT (0x7430): Timer 0 Counter Register

P_TMR1_TCNT (0x7431): Timer 1 Counter Register

P_TMR2_TCNT (0x7432): Timer 2 Counter Register

P_TMR3_TCNT (0x7433): Timer 3 Counter Register

P_TMR4_TCNT (0x7434): Timer 4 Counter Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
TMRCNT							

P_TMR0_TGRA (0x7440): Timer 0 General Register A

P_TMR0_TGRB (0x7441): Timer 0 General Register B

P_TMR0_TGRC (0x7442): Timer 0 General Register C

P_TMR1_TGRA (0x7440): Timer 1 General Register A

P_TMR1_TGRB (0x7441): Timer 1 General Register B

P_TMR1_TGRC (0x7442): Timer 1 General Register C

P_TMR2_TGRA (0x7440): Timer 2 General Register A

P_TMR2_TGRB (0x7441): Timer 2 General Register B

P_TMR2_TGRC (0x7442): Timer 2 General Register C

P_TMR3_TGRA (0x7440): Timer 3 General Register A

P_TMR3_TGRB (0x7441): Timer 3 General Register B

P_TMR3_TGRC (0x7442): Timer 3 General Register C

P_TMR4_TGRA (0x7440): Timer 4 General Register A

P_TMR4_TGRB (0x7441): Timer 4 General Register B

P_TMR4_TGRC (0x7442): Timer 4 General Register C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRGLR							

P_TMR0_TPR (0x7435): Timer 0 Period Register

P_TMR1_TPR (0x7436): Timer 1 Period Register

P_TMR2_TPR (0x7437): Timer 2 Period Register

P_TMR3_TPR (0x7438): Timer 3 Period Register

P_TMR4_TPR (0x7439): Timer 4 Period Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
1	1	1	1	1	1	1	1
TMRPRD							

P_TMR0_TBRA (0x7450): Timer 0 Buffer Register A

P_TMR0_TBRB (0x7451): Timer 0 Buffer Register B

P_TMR0_TBRC (0x7452): Timer 0 Buffer Register C

P_TMR1_TBRA (0x7453): Timer 1 Buffer Register A

P_TMR1_TBRB (0x7454): Timer 1 Buffer Register B

P_TMR1_TBRC (0x7455): Timer 1 Buffer Register C

P_TMR2_TBRA (0x7456): Timer 2 Buffer Register A

P_TMR2_TBRB (0x7457): Timer 2 Buffer Register B

P_TMR3_TBRA (0x7458): Timer 3 Buffer Register A

P_TMR3_TBRB (0x7459): Timer 3 Buffer Register B

P_TMR3_TBRC (0x745A): Timer 3 Buffer Register C

P_TMR4_TBRA (0x745C): Timer 4 Buffer Register A

P_TMR4_TBRB (0x745D): Timer 4 Buffer Register B

P_TMR4_TBRC (0x745E): Timer 4 Buffer Register C

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TMRBUF							

P_TMR0_IOCctrl (0x7410) : Timer 0 IO control register

P_TMR1_IOCctrl (0x7411) : Timer 1 IO control register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved						IOCMODE	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE						IOAMODE	

P_TMR2_IOCctrl (0x7412) : Timer 2 IO control register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE						IOAMODE	

P_TMR3_IOCctrl (0x7413) : Timer 3 IO control register

P_TMR4_IOCctrl (0x7414) : Timer 4 IO control register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved				IOCMODE			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
IOBMODE				IOAMODE			

P_TMR0_INT (0x7420): Timer 0 Interrupt Enable Register

P_TMR1_INT (0x7421): Timer 1 Interrupt Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
Reserved						PDCIE	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	TCUIE	TCVIE	TPRIE	Reserved	TGCIE	TGBIE	TGAIE

P_TMR2_INT (0x7422): Timer 2 Interrupt Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TADSE	Reserved		TPRIE	Reserved		TGBIE	TGAIE

P_TMR3_INT (0x7423): Timer 3 Interrupt Enable Register

P_TMR4_INT (0x7424): Timer 4 Interrupt Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TADSE	Reserved		TPRIE	TGDI	Reserved		

P_TMR0_Status (0x7425): Timer 0 Interrupt Status Register

P_TMR1_Status (0x7426): Timer 1 Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	TCUIF	TCVIF	TPRIF	Reserved	TGCIF	TGBIF	TGAIF

P_TMR2_Status (0x7427): Timer 2 Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R	R	R/W	R/W
0	0	0	0	0	0	0	0
TCDF	Reserved		TPRIF	Reserved		TGBIF	TGAIF

P_TMR3_Status (0x7428): Timer 3 Interrupt Status Register
P_TMR4_Status (0x7429): Timer 4 Interrupt Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R/W	R/W	R	R	R
0	0	0	0	0	0	0	0
TCDF	Reserved		TPRIF	TGdif	Reserved		

P_TMR_Start (0x7405): Timer Counter Start Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved		TMR4ST	TMR3ST	TMR2ST	TMR1ST	TMR0ST	

P_TMR_Output (0x7406): Timer Output Enable Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved		TMR4FOE	TMR4EOE	TMR4DOE	TMR4COE	TMR4BOE	TMR4AOE

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved		TMR3FOE	TMR3EOE	TMR3DOE	TMR3COE	TMR3BOE	TMR3AOE

P_TMR_LDOK (0x740A) : Timer Load-OK Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TLDCHK						LDOKE1	LDOKE0

P_TMR3_OutputCtrl (0x7407): Timer 3 Output Control Register
P_TMR4_OutputCtrl (0x7408): Timer 4 Output Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
DUTYMODE	POLP	Reserved			WPWM	VPWM	UPWM

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SYNC		WOC		VOC		UOC	

P_TPWM_Write (0x7409): Timer/PWM Module Write Enable Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
Reserved						TMR4WE	TMR3WE

P_POS0_DectCtrl (0x7462): Timer 0 Position Detection Control Register

P_POS1_DectCtrl (0x7463): Timer 1 Position Detection Control Register

B15	B14	B13	B12	B11	B10	B9	B8		
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W		
0	0	0	0	0	0	0	0		
SPLCK		SPLMOD		SPLCNT					

B7	B6	B5	B4	B3	B2	B1	B0	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	
PDEN		SPDLY						

P_POS0_DectData (0x7464): Timer 0 Position Detection Data Register

P_POS1_DectData (0x7465): Timer 1 Position Detection Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved				PDR			

P_TMR3_DeadTime (0x7460): Timer 3 Dead Time and Control Register

P_TMR4_DeadTime (0x7461): Timer 4 Dead Time and Control Register

B15	B14	B13	B12	B11	B10	B9	B8	
R	R/W	RW	R/W	R	R	R	R	
0	0	0	0	0	0	0	0	
Reserved	DTWE	DTVE	DTUE	Reserved				

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W						
0	0	0	0	0	0	0	0
Reserved	DTP						

P_Fault1_Ctrl (0x7466): Fault Input 1 Control and Status Register

P_Fault2_Ctrl (0x7467): Fault input 2 Control and Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R	R	R	R
0	0	0	0	0	0	0	0
OCE	OCIE	OCLS	OSF	Reserved			

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
FTPINE	FTPINIE	FTPINIF	Reserved	FTCNT			

P_OL1_Ctrl(0x7468): Overload input 1 Control and Status Register

P_OL2_Ctrl(0x7469): Overload input 2 Control and Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	RW	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLEN	CNTSP	OLMD		OLST	RTTMB	RTPWM	RTOL

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
OLIE	OLIF	Reserved		OLCNT			

P_Fault1_Release(0x746A): Fault 1 Flag Release Register

P_Fault2_Release(0x746B): Fault 2 Flag Release Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
FTRR							

20.7 Programmable Registers of CMT Timer on SPMC75X Family MCU

P_CMT_Start(0x7500): Compare Match Timer Start Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
Reserved						ST1	ST0

P_CMT_Ctrl(0x7500): Compare Match Timer Control and Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM1IF	CM1IE	Reserved			CKB		

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CM0IF	CM0IE	Reserved			CKA		

P_CMT0_TCNT(0x7508): Compare Match Timer 0 Counter Register

P_CMT1_TCNT(0x7509): Compare Match Timer 1 Counter Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
CMTCNT							

P_CMT0_TPR(0x7510): Compare Match Timer 0 Period Register

P_CMT1_TPR(0x7511): Compare Match Timer 1 Period Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
CMTPR							

20.8 Programmable Registers of TMB / Buzzer Unit on SPMC75X Family MCU

P_TMB_Reset(0x70B8): Time Base Reset Register

B15	B14	B13	B12	B11	B10	B9	B8
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0
TBRR							

P_BZO_Ctrl(0x70B9): Buzzer Output Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
BZOEN	Reserved						

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0
Reserved							BZOCK

20.9 Programmable Registers of SPI on SPMC75X Family MCU

P_SPI_Ctrl(0x7140): SPI Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIE	Reserved			SPIRST	SPISPCLK		SPIMS

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
Reserved		SPIPHA	SPIPOL	SPISMPS	SPIFS		

P_SPI_TxStatus(0x7141): SPI Transmit Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R	R	R	R	R
0	0	0	0	0	0	0	0
SPITXIF	SPITXIE	SPITXBF	Reserved				

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

P_SPI_TxBuf(0x7142): SPI Transmit Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPITXBUF							

P_SPI_RxStatus(0x7143): SPI Receive Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R/W	R	R
0	0	0	0	0	0	0	0
SPIRXIF	SPIRXIE	Reserved			FERR	Reserved	

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

P_SPI_RxBuf(0x7144): SPI Receive Buffer Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
SPIRXBUF							

20.10 Programmable Registers of UART on SPMC75X Family MCU

P_UART_Data(0x7100): UART Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved			OE	Reserved		FE	PE

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTDATA							

P_UART_RXStatus(0x7101): UART Reception Error Flag Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	Bb2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved				OE	Reserved	FE	PE

P_UART_Ctrl(0x7102): UART Reception Error Flag Register

B15	B14	B3	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	W	R/W	R/W	R
0	0	0	0	0	0	0	0
RXIE	TXIE	Reserved	UEN	Reset	TXCHSEL	RXCHSEL	Reserved

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R/W	R/W	R/W	R
0	0	0	0	0	0	0	0
Reserved				SBSEL	PSEL	PEN	Reserved

P_UART_BaudRate(0x7103): UART Baud Rate Setup Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
UARTBUD							

P_UART_Status(0x7104): UART Status Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
RXIF	TXIF	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved	RXBF	Reserved		BY	Reserved		

20.11 Programmable Registers of ADC on SPMC75X Family MCU

P_ADC_Setup(0x7160): ADC Setup Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
1	0	0	0	0	0	0	0
ADCCS	ADCEN	Reserved			ADCFCS	ADCEXTRG	

B7	B6	B5	B4	B3	B2	B1	B0
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ASPEN	Reserved						

P_ADC_Ctrl(0x7161): ADC Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCIF	ADCIE	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
R	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0
ADCRDY	ADCSTR	Reserved			ADCCHS		

P_ADC_Channel(0x7166): ADC Input Channel Select Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
Reserved							

B7	B6	B5	B4	B3	B2	B1	B0
R/W							
0	0	0	0	0	0	0	0
ADCCH7	ADCCH6	ADCCH5	ADCCH4	ADCCH3	ADCCH2	ADCCH1	ADCCH0

P_ADC_Data(0x7166): ADC Data Register

B15	B14	B13	B12	B11	B10	B9	B8
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCDATA							

B7	B6	B5	B4	B3	B2	B1	B0
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
ADCDATA		Reserved					

20.12 Programmable Registers of Watchdog Timer on SPMC75X Family MCU
P_WatchDog_Ctrl(0x700A): WatchDog Control Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R	R	R	R	R	R
0	0	0	0	0	0	0	0
WDEN	WDRS	Reserved					

B7	B6	B5	B4	B3	B2	B1	B0
W	W	W	W	W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDCHK					WDPS		

P_WatchDog_Clr(0x700B): WatchDog Clear Register

B15	B14	B13	B12	B11	B10	B9	B8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDTCLR							

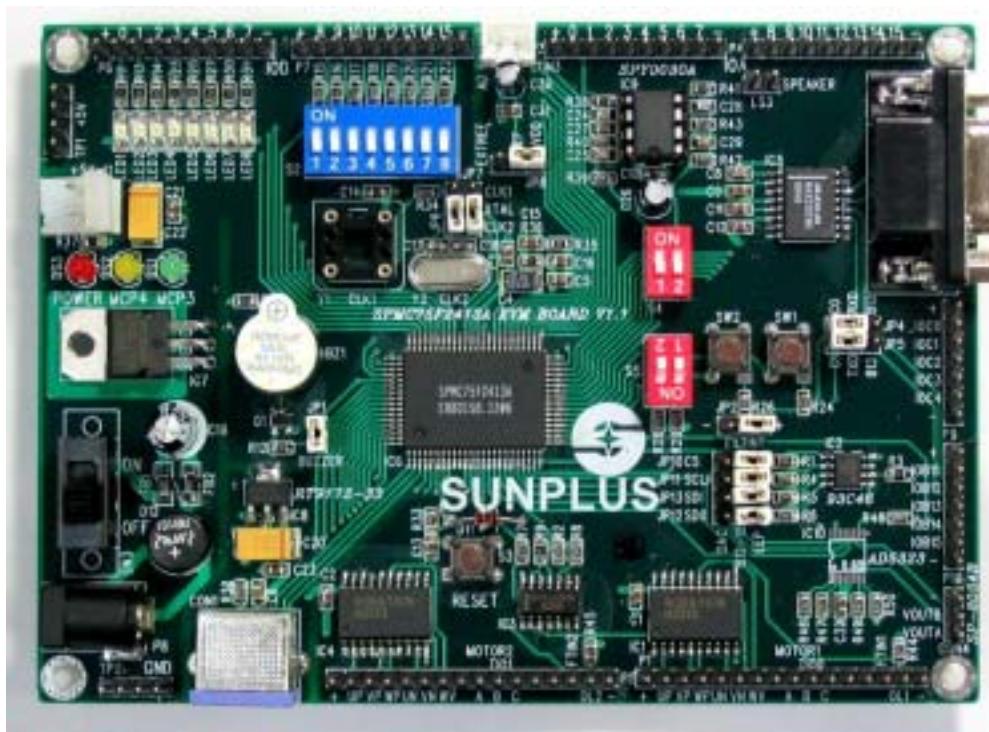
B7	B6	B5	B4	B3	B2	B1	B0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0
WDTCLR							

21 SPMC75X Family MCU Development System

21.1 Development system Setup diagram

Connection: PC parallel port / USB port → SUNPLUS Probe → EVM board

21.2 SPMC75F2413A EVM board v1.1



21.3 SPMC75F2413A EVM board v1.1 schematics

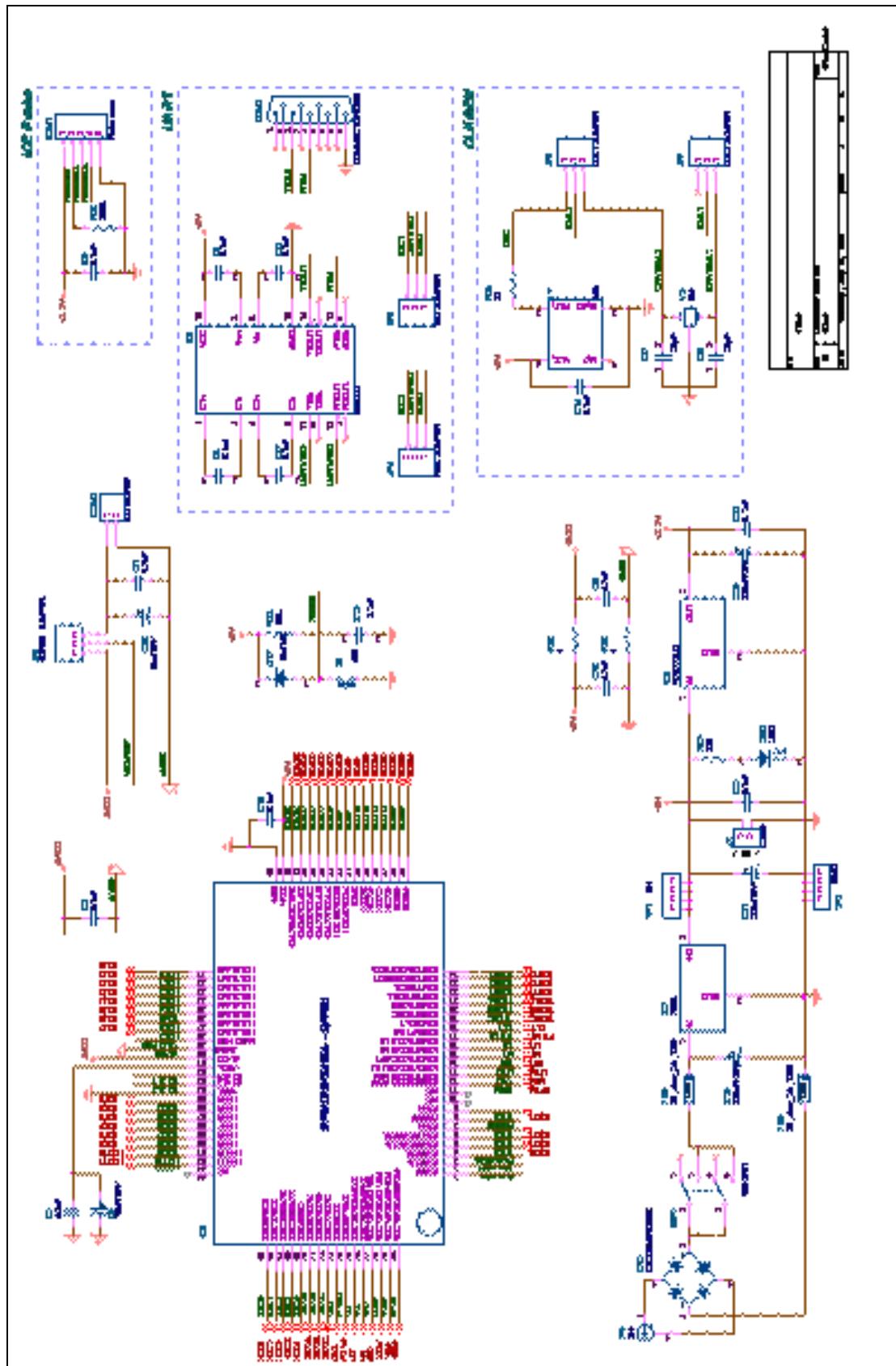


Figure 21-1 SPMC75F2413A EVM Board Schematic – Part 1

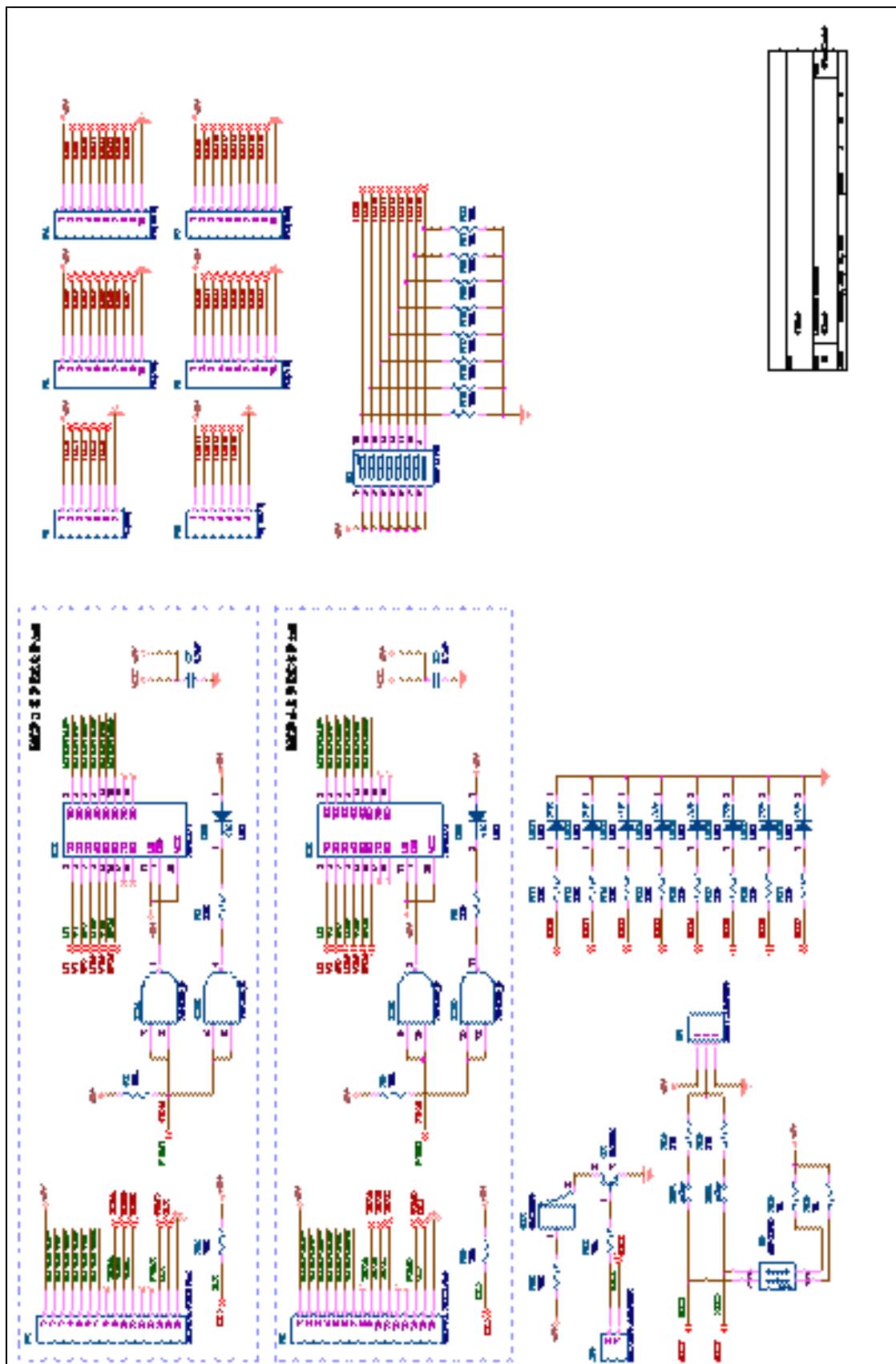


Figure 21-2 SPMC75F2413A EVM Board Schematic – Part 2

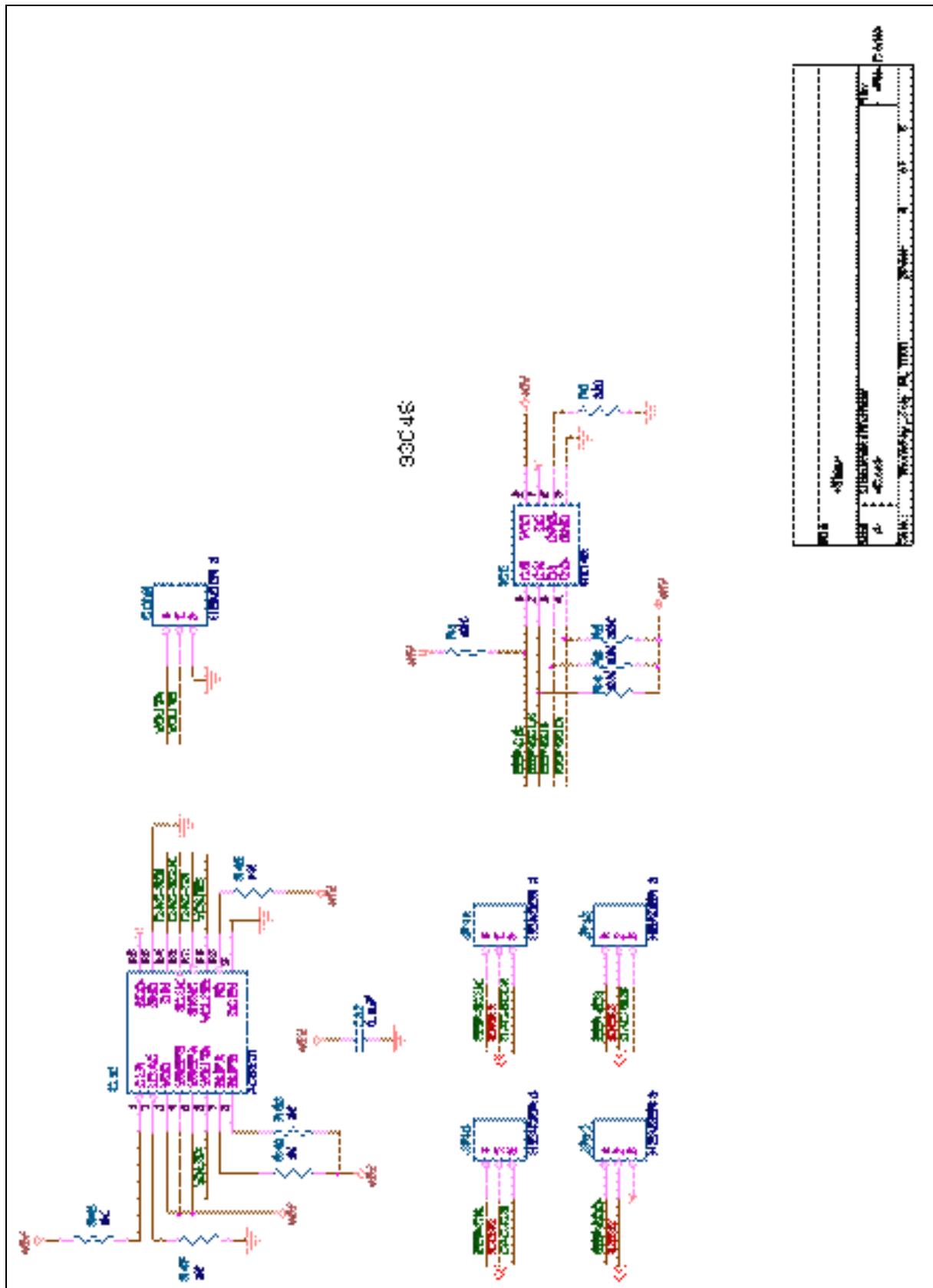


Figure 21-3 SPMC75F2413A EVM Board Schematic – Part 3

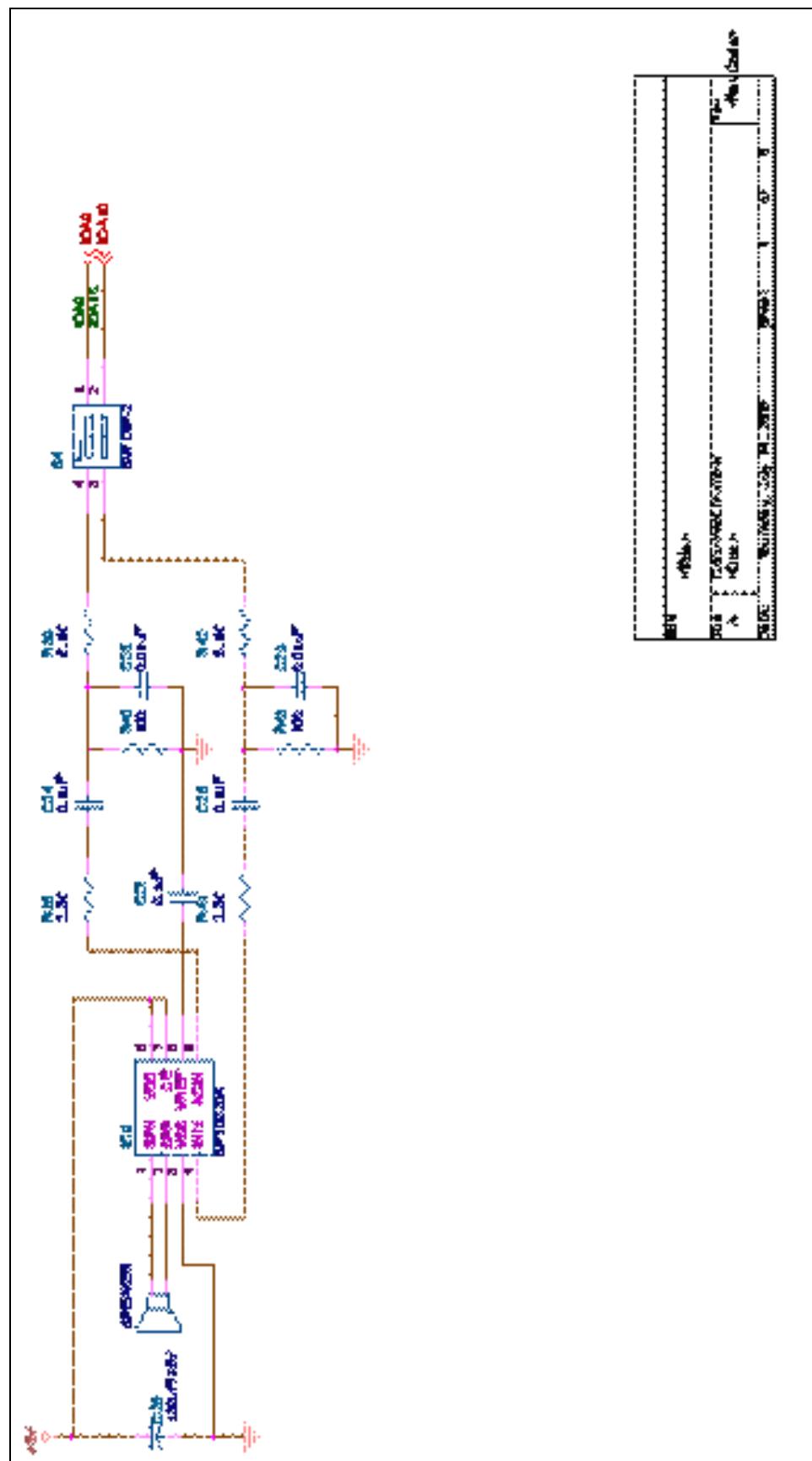


Figure 21-4 SPMC75F2413A EVM Board Schematic – Part 4

22 SUNPLUS µnSP IDE Development Environment

22.1 General Description

Welcome to the SUNPLUS µnSP® IDE, a 16-bit Integrated Development Environment (IDE), own fully integrated Windows development tool. Through µnSP® IDE, editing, compiling, linking, debugging and simulating functions can be used on your own way. Its user-friendly interface, pull-down menu, short-cut command, and fast-access command list assist users to write and debug program on SUNPLUS µnSP® IDE easily and efficiently. A simulation mode is also included to simulate program functions without connecting to a device emulation board.

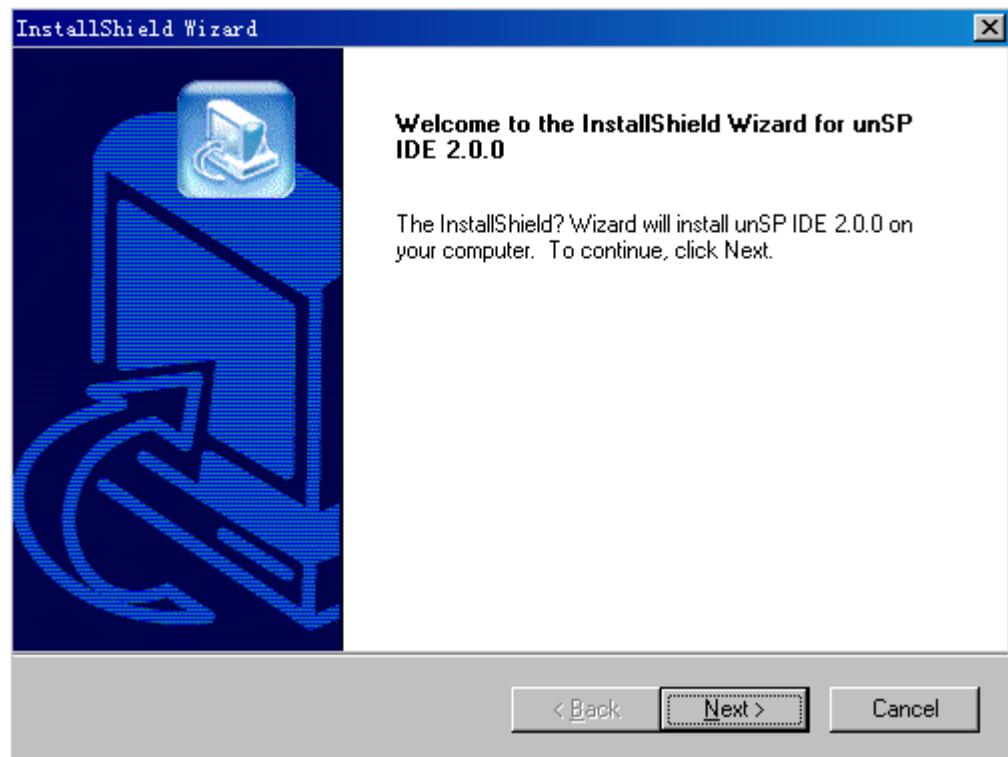
This documentation will guide you through the µnSP® IDE installation, quick start, command and function usage. For individual product specification, please contact your SUNPLUS sales representatives for further request.

22.2 Installation

This section introduces how to install µnSP® IDE and common on your computer. The IDE can be run on Windows98®, Windows2000® and WindowsXP®.

Note: You must have Administrator privilege when install the IDE on Win2000/XP. The printer-port mode of "SPP" must be set in BIOS setting, and hardware address is recommended to be set to 378H as well. If you use USB probe, plug in USB probe after InstallShield Wizard completes.

1. Run the install file on your system. After extract all files, a welcome screen will be displayed.



2. Follow the on-screen instructions and µ'nSP® IDE and common will be installed on your computer.
3. Select [Start Menu]→[Program]→[sunplus]→[unSPIDE]→[unSPIDE] to run the IDE.
4. After install µ'nSP® IDE setup program, if you want to use USB probe, please plug in USB probe. Windows system will install USB probe driver automatically.
5. In Windows XP, if you plug-in USB probe, Windows system will ask you to continue installing driver or not. Please click [Continue(C)], then, system will install USB probe driver automatically.

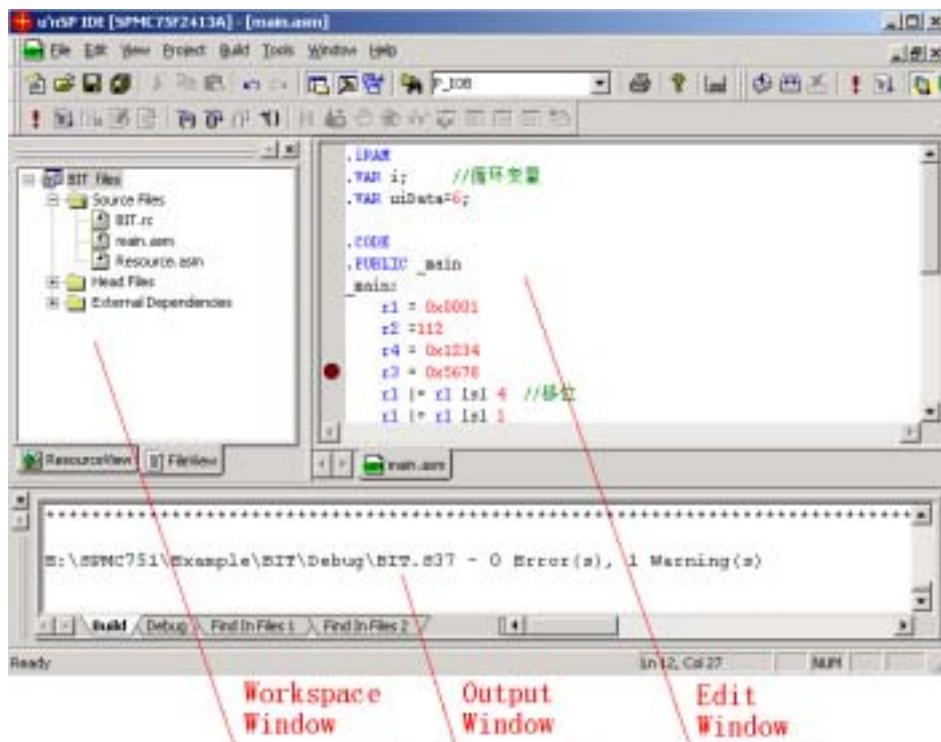
22.3 Quick Start

1. Run μ nSP® IDE from Start menu.
2. Click [File]→[Open Project].
3. The project window is located at the left side. You can open source files from FileView window by double clicking on the filename.
4. Click [Build]→[Rebuild All] to compile and link source program. If there is no error, continue the followed steps.
5. Click [Build]→[Start Debug]→[Download] to download a program, then use the menu commands in Debug menu to run and debug it. You can use [Build]→[Start Debug]→[Go] to run the program directly.

22.4 Get to Know The μ nSP® IDE

22.4.1 Main Window

μ nSP® IDE includes a series of integrated development tools that can be used to edit, compile, link and debug your program within IDE. Users are allowed to open more than one window at the same time on the screen. There are three major windows you will see in μ nSP® IDE, Workspace Window, Output Window and Edit Window. You can switch among these windows by simply click within window area.



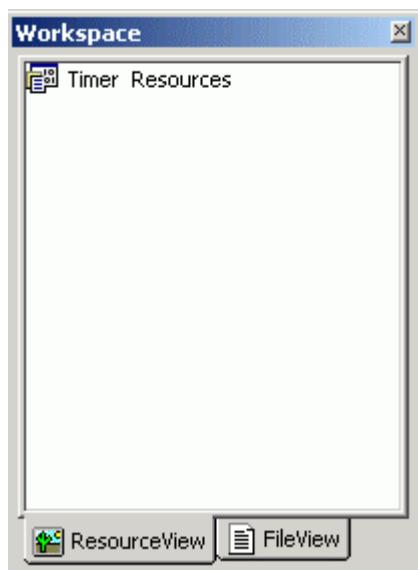
22.4.2 Workspace Window

Workspace displays file information. The Workspace window lists all files and their relation in current project, called ResourceView and FileView. ResourceView window displays all resource files used in current project. FileView window shows logical relationships among all files in current project. You are able to switch them by clicking the label of ResourceView and FileView at the bottom of the Workspace Window.

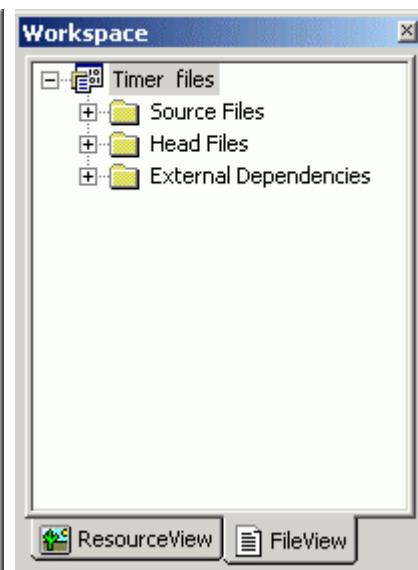
NOTE:

- (1) Workspace window does not represent files' physical location on your hard disk.
- (2) Resource Table created by IDE will according to the order of the resource tree.
- (3) You can arrange the resource order by drag and drop with mouse.

<ResourceView>

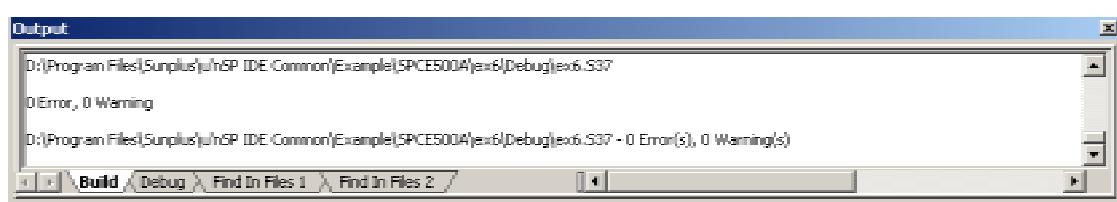


<FileView>



22.4.3 Output Window

Output window shows the result of compilation, linkage, debug and find.



Build: Show the information of compilation and linkage. Errors and warnings may be given during compilation and linkage. A program is built completely and successfully when no error occurred in the report.

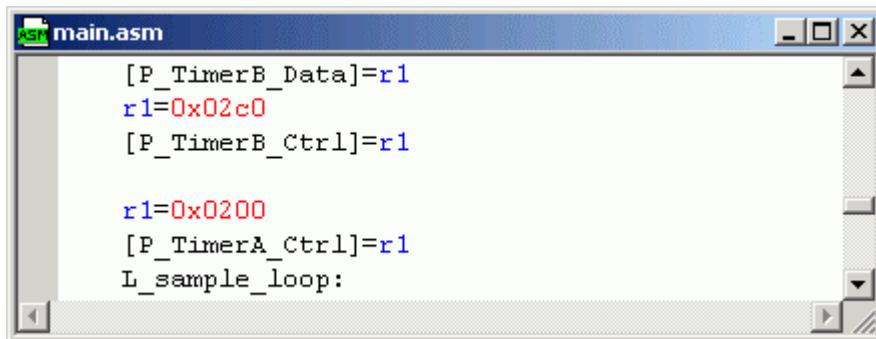
Debug: Show the debug message.

Find in Files: Show the result of finding text in files.

22.4.4 Edit Window

Text Editor

Use Text Editor to edit a program. When you open a file from a project, all the contents in a file are displayed in the text editor.

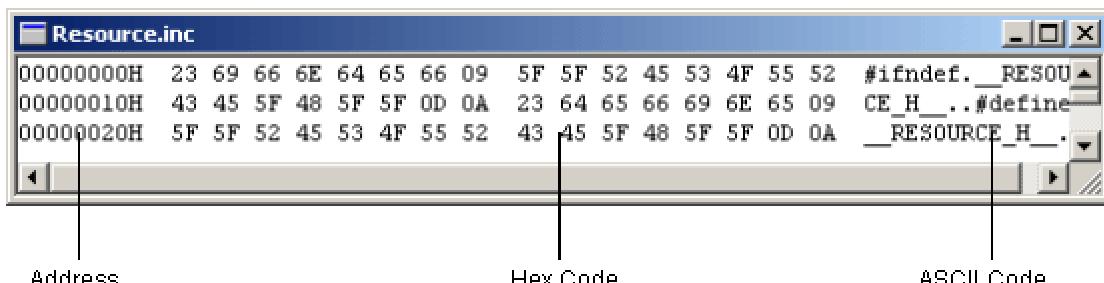


```
main.asm
[P_TimerB_Data]=r1
r1=0x02c0
[P_TimerB_Ctrl]=r1

r1=0x0200
[P_TimerA_Ctrl]=r1
L_sample_loop:
```

22.4.5 Binary Editor

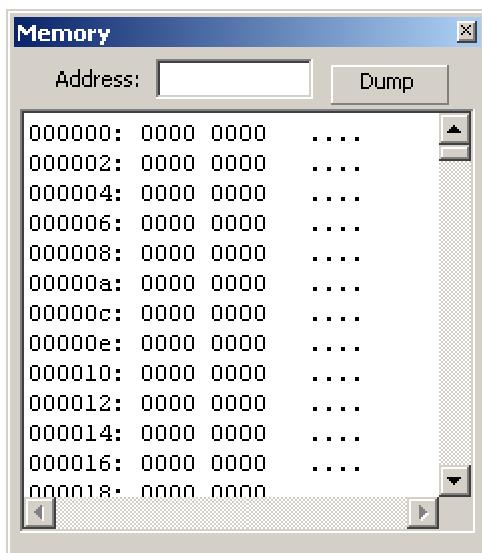
Use Binary Editor to edit an existing resource file at the binary level in either hexadecimal or ASCII format.



Address	Hex Code	ASCII Code
00000000H	23 69 66 6E 64 65 66 09 5F 5F 52 45 53 4F 55 52	#ifndef. _RESOU
00000010H	43 45 5F 48 5F 5F 0D 0A 23 64 65 66 69 6E 65 09	CE_H...#define
00000020H	5F 5F 52 45 53 4F 55 52 43 45 5F 48 5F 5F 0D 0A	_RESOURCE_H_.

22.4.6 Debug Window

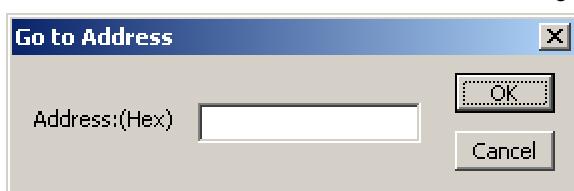
Memory Window



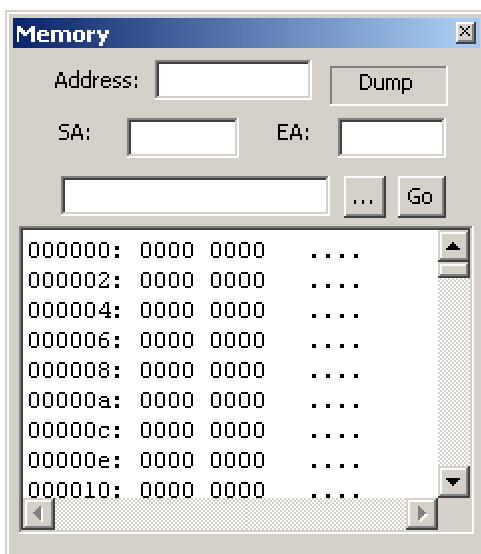
While debugging, you can view memory content by selecting [View]→[Debug Windows]→[Memory].

Memory window displays numbers in hexadecimal format. You are allowed to view memory content at specified location by typing in Address text box. Then, the memory window displays the contents of memory locations beginning at the address specified in the Address(0x0000000 by default). You may also view another block of data by clicking on the scroll bar. In addition, you can also modify the content of specified address unit by just clicking the location you want to modify and key in the content.

Double click on the window, the Go to Address dialog box will be popped up:

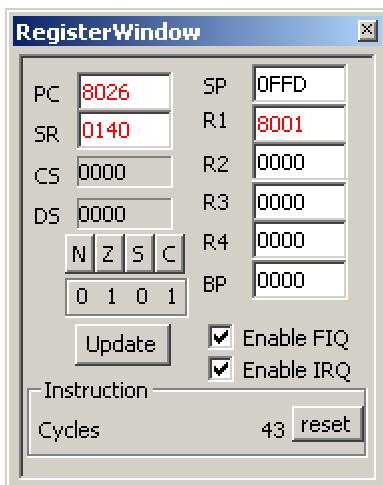


Key in the address you want and the memory window will display the memory data from the address you specified.



On the upper window, a Dump function, which can be used to dump a block of memory contents to one file, is provided. You can type data in SA and EA box to specify the memory range. Click [Go] to dump memory to the file you specified.

22.4.7 Register Window



Current contents of the general registers and CPU status registers can be checked and changed in this window.

PC: Program counter

SR: Status register

CS: Code segment

DS: Data segment

SP: Stack pointer

R1 ~ R4: General registers

BP: Base pointer

N, Z, S, C: Negative, Zero, Sign, Carry

Cycles: Total cycle count after last count reset to current PC location

Enable FIQ/IRQ: Enable or disable FIQ/IRQ

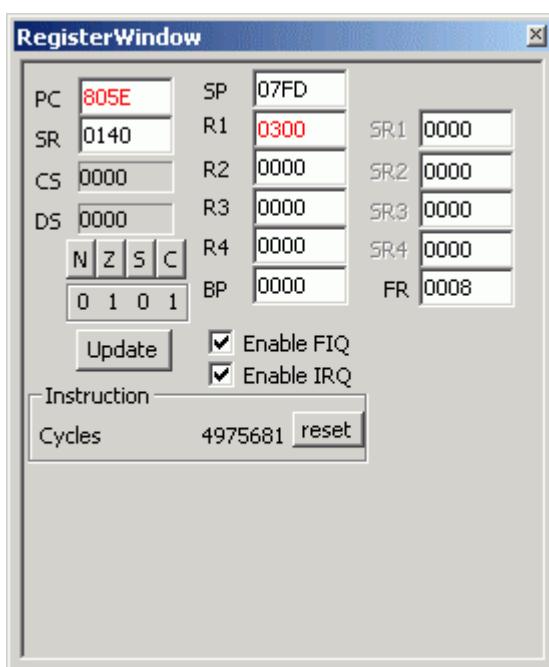
[Reset]: Reset the configuration

[Update]: After you modify the flag and enable the interrupt, press it to update the configuration

Note:

In ICE mode, you can not see the instruction cycle counter.

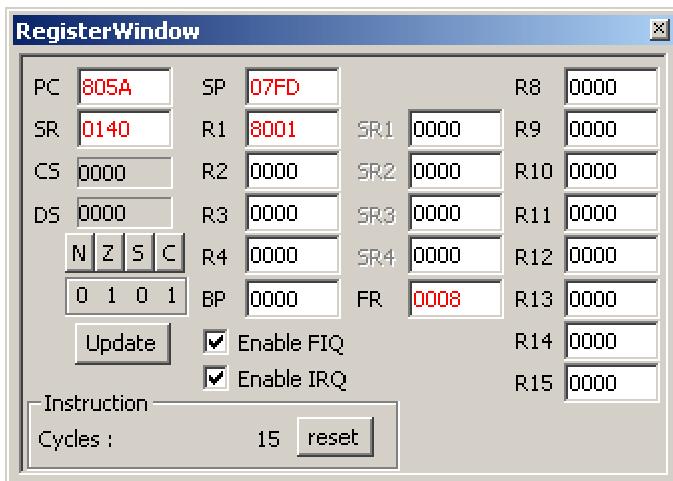
For the ISA1.2 instruction set, there are additional five registers.



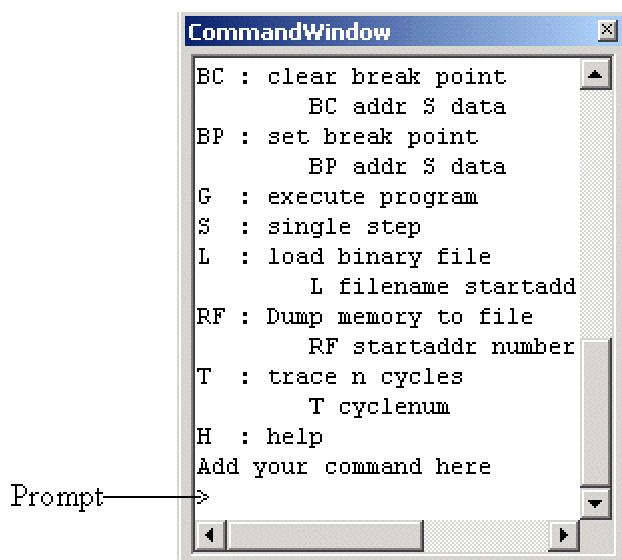
SR1~SR4: Secondary register for R1~R4.

FR: Flag register

For the ISA2.0 instruction set, there are additional eight registers.



22.4.8 Command Window



Click [View]→[Debug Windows]→[CommandWindow] to open command window. The on-line help can be called by typing "H" in text box within command window. A list of all commands and syntax are listed. Users can type in a command in the text box to run a specified command.

Command:

- | | |
|-------|-----------------------------------|
| Q : | Quit µ'nSP® IDE |
| Dump: | Dump memory content |
| | Syntax: Dump startaddr number |
| | Ex: Dump memory for 0x100 ~ 0x1ff |
| | > dump 100 100 |
| EF: | EnableFIQ flag(Default) |
| DF: | Disable FIQ flag |

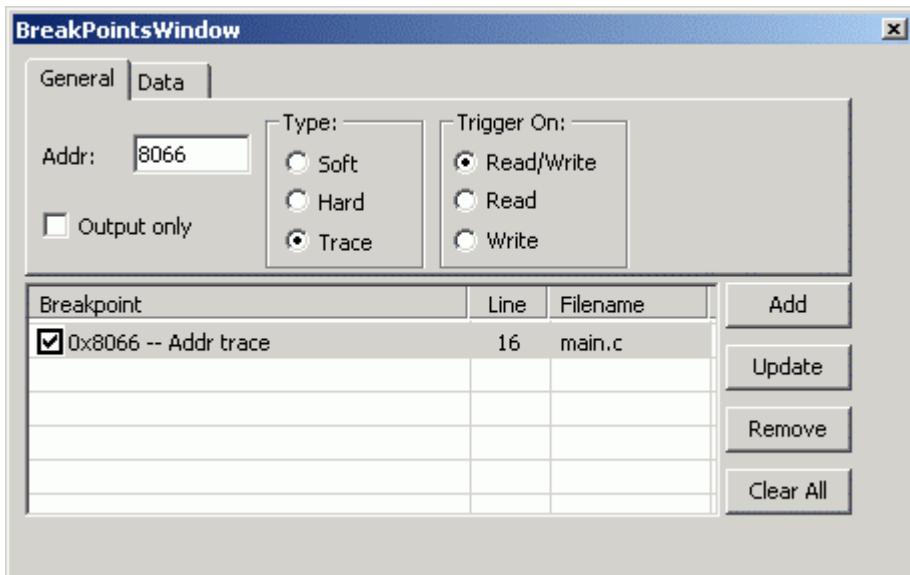
Command:

EI:	Enable IRQ flag (Default)
DI:	Disable IRQ flag
SN:	Set negative flag
NN:	Clear negative flag
SS:	Set sign flag
NS:	Clear sign flag
SZ:	Set zero flag
NZ:	Clear zero flag
SC:	Set carry flag
NC:	Clear carry flag
X:	Reset program
RX:	Set register value Syntax: RX n value Ex: Change R3 value to 0xabcd > RX 3 abcd
O:	Set RAM value Syntax: O addr value Ex: Change memory 0x7016 to 0xabcd > O 7016 abcd
F:	Fill a block of ram with specific value Syntax: F startaddr number value Ex: Fill ram 0x100 - 0x1ff with 0x1234 > F 100 100 1234
BC:	Clear breakpoint Syntax:BC addr S data Ex:Clear a breakpoint at 0x8000, BPS flag:0x00, Data:0x1234 >BC 8000 0 1234 *it only clear the breakpoint match these condition
BP:	Set breakpoint Syntax: BP addr S data Ex: Set a breakpoint at 0x8000, BPS flag:0x00, Data:0x1234 >BP 8000 0 1234
G:	Execute program
S:	Execute single step function
L:	Load binary file into memory Syntax: L filename startaddr endaddr RAMaddr If you want to load all files into memory, set the start and end to 0. Ex: Load 0x100 - 0x1ff of file to memory address 0x8000 > L test.bin 100 1ff 8000

Command:

- RF: Dump memory to file
Syntax: RF startaddr number filename
Ex: Dump memory 0x100 - 0x1ff to file test.bin
> RF 100 100 test.bin
- T Trace n cycle
Syntax: T cyclenum
Ex: Trace 5
- H: Show the help context

22.4.9 Breakpoint Window



Use the Breakpoints window, you can set, remove, enable, disable and view breakpoints. The breakpoints you set will be saved as a part of your project.

Addr:

The address of breakpoint.

Type:

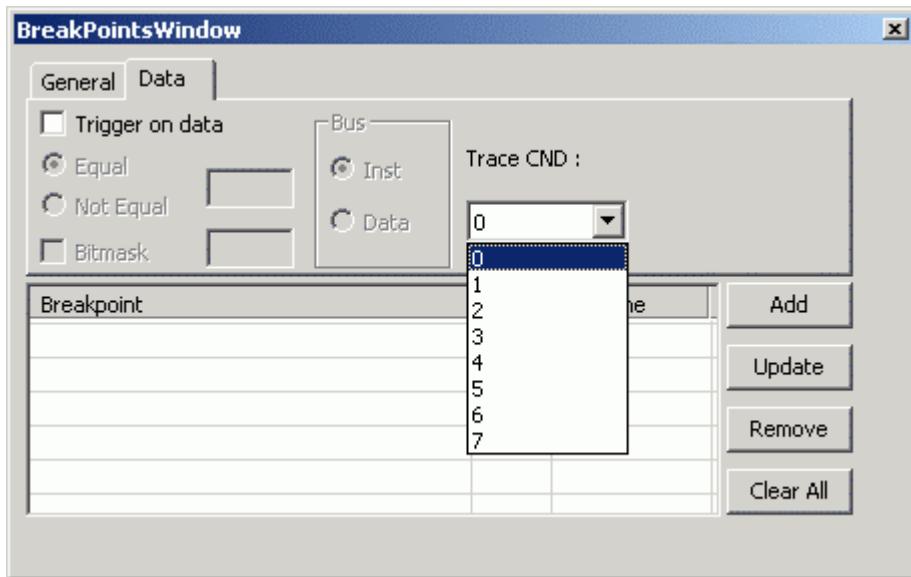
The breakpoint type. If ICE mode is selected, the Hard item is enabled, the Soft item is enabled if the instruction set is ISA 1.2 and above. For simulator, only Soft item is enabled. When trace breakpoint is matched, the data in the tracebuffer will be fetched and the program will go on to run.

Output Only:

Output a pulse signal in ICE mode to certain pin of chip.

Trigger on Write&Read:

When access data in breakpoint address, the interrupt is toggled on.



Trigger on Data:

In ICE mode, if you select Hard item in General page, the Trigger on Data will be enabled. If Trigger on Data and Equal(Not Equal) is selected, application will check whether the data in breakpoint address is equal(not equal) to the data specified in Trigger on Data text box. If equal(not equal) is matched, program will stop execution.

Bitmask:

Mask certain data bit which is in breakpoint address. Not Equal item is enabled if Bitmask is enabled.

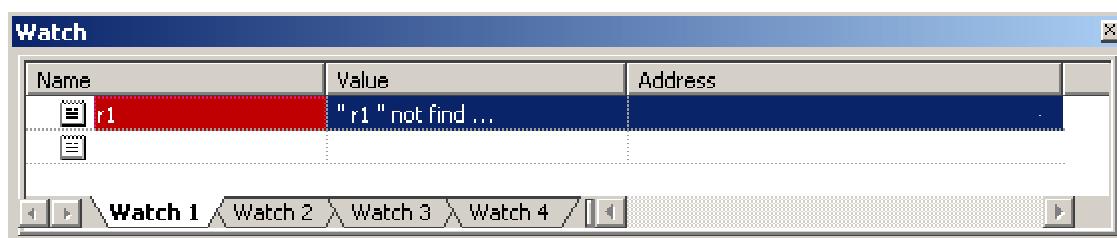
Bus:

In ISA2.0 version and above, select Inst for instruction type/select Data for data type.

Trace CND:

If 1~7 is selected, the relevant CND pin will output data during program running.

22.4.10 Watch Window



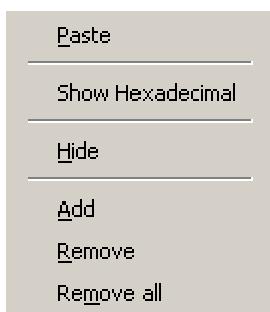
Watch window is used to specify variables that you want to watch while debugging program. You can also

modify the value of a variable in this window.

The Watch window contains four tabs: Watch1, Watch2, Watch3, and Watch4. Each tab displays a user-specified list of variables with a spreadsheet.

If the value of a variable appears in red, it indicates that the value has recently changed during program running.

By right click on this window, a context bar features Paste, Show decimal/hexadecimal, Hide, Add, Remove and Remove all functions can be toggled on.



Paste: Ease from typing.

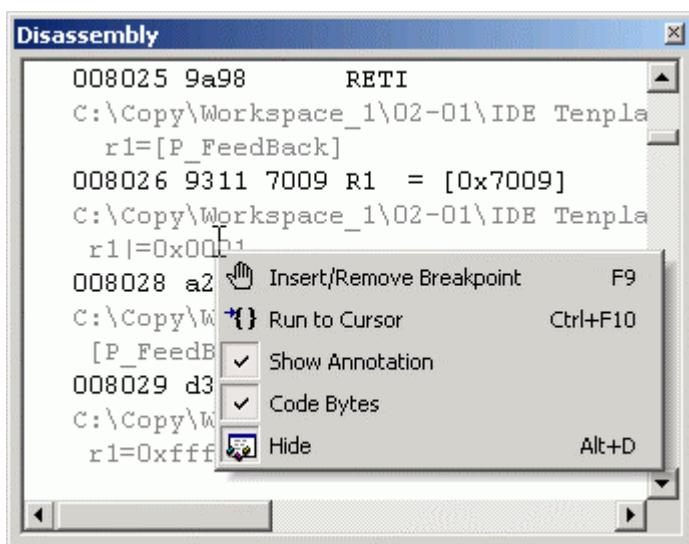
Show decimal/hexadecimal: Show variable value in decimal/hexadecimal.

Hide: Hide Watch window.

Add/Remove/Remove all: Edit current list in WatchN(N=1,2,3,4).

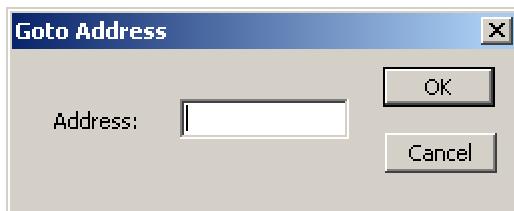
You can select Array Size in Debug page to specify the array length in Watch window. When Color red for changed value is enabled, you can find the changed value is displayed in red in Watch window, but this option can affect the debug speed.

Disassembly Window



The Disassembly window can be used for debugging code. The contents of disassembly will be changed with PC address.

Double click on the window, you are able to go to a specific address of memory . In the pop-up window, enter the address and select [OK], the disassembly line of that address will be displayed on the top of the window.



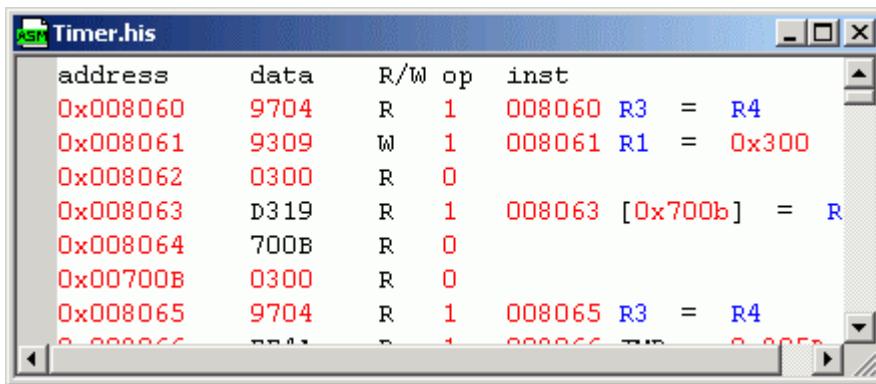
By right clicking, the context menu will be enabled. You can check annotation by selecting [Show Annotation] after right clicking or selecting [Tools]→[Option]→[Debug]→[Source annotation]. Op code can be shown after you select [Code Bytes] on hotkey or select [Tools]→[Option]→[Debug]→[Code bytes].

You can insert, remove breakpoint, run program to current cursor line, view source code and operand, hide the Disassembly window.

History Buffer Window

The executed instructions, status and memory contents will be saved in history buffer when program is running(be sure TraceBuffer is enable when you configure the project).

Click  from the debug toolbar, a History buffer window is shown as follows:



address	data	R/W	op	inst
0x008060	9704	R	1	008060 R3 = R4
0x008061	9309	W	1	008061 R1 = 0x300
0x008062	0300	R	0	
0x008063	D319	R	1	008063 [0x700b] = R
0x008064	700B	R	0	
0x00700B	0300	R	0	
0x008065	9704	R	1	008065 R3 = R4

If you select "Save Instruction Only" when you set project option, only the instruction will be shown in the window.

22.5 Project

22.5.1 Project Window

In project window, you will see the following information.

1. Source file name and logical relationship. In project window, the directory-tree describes the logical relationships among files. It does not reflect physical position on your hard disk. Double click on a filename, it will be opened on the Edit Window.
2. Tool configuration for current program, for example, the option settings on compiler and linker. Click [Project]→[Setting] to set up a project options.

Use Project

You can add and move files in project and update the relationship among the files in current project.

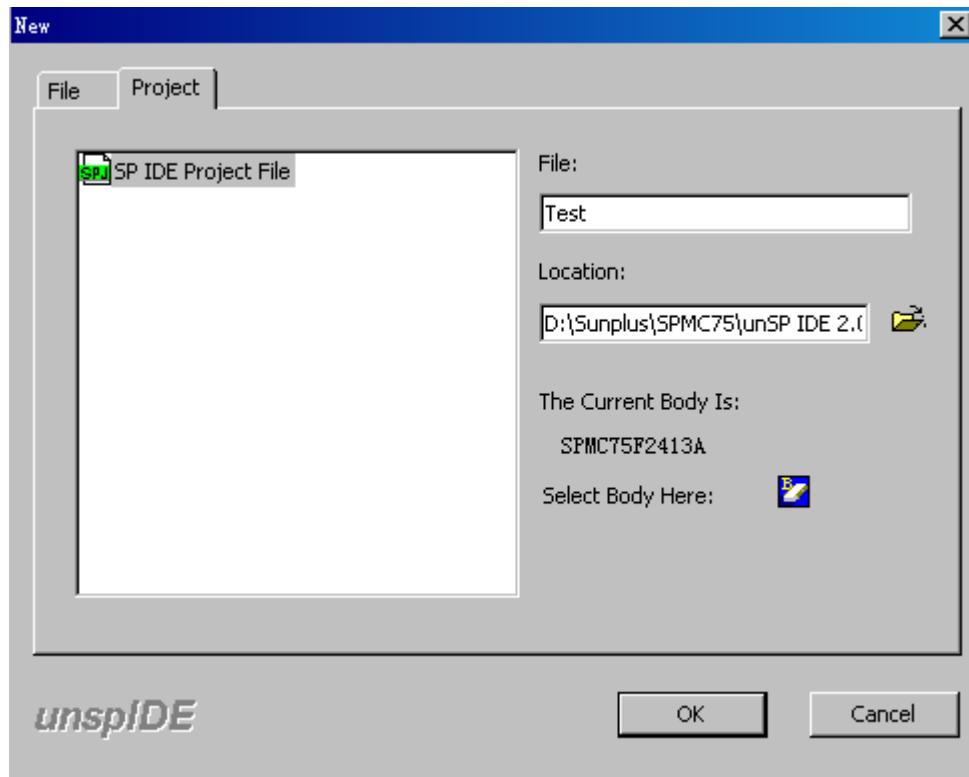
File in New Project

A new project contains five types of files:

3. Project file (.spj): The file extension is 'spj', which contains all commands and options to create a project.
4. Resource file (.rc): The file extension is 'rc', which contains all resource files in current project.
Note: You cannot modify .spj and .rc files directly.
5. Resource table file(.asm): The resource table is created according to the order of the resource tree.
6. Resource table head file(.inc)
7. Make file(Makefile)
8. C included file(.h)

22.5.2 Create Project

1. Click [File]→[New] to display a "New" dialog box.

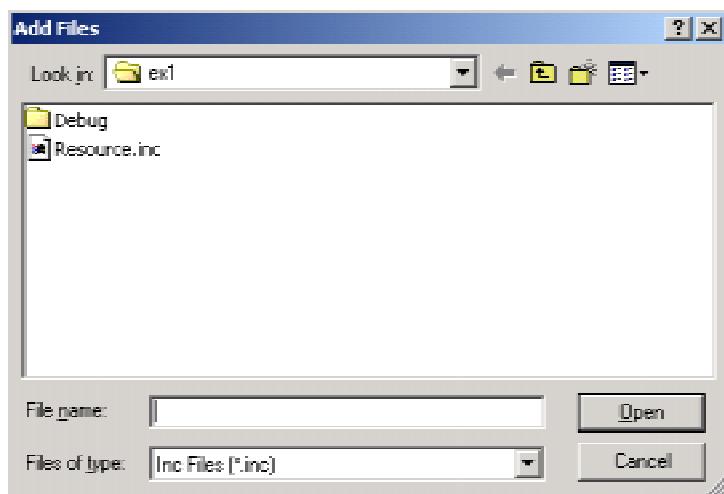


2. Click Project label.
3. Type a project name in file box.
4. Browse and select path, or input the location directly to specify a path for your project.
5. Select a body type.
6. Click [OK].

After creating a project, the µ'nSP® IDE will automatically generate the following files:.spj(project file) .rc (resource file), .h file (C included file), Resource.asm (Resource Table), Resource.inc (Resource Table Head file), and Makefile for the project. For example, if a project, named "test" is created, the µ'nSP® IDE will automatically generate a test directory with "test.rc" file, test.spj, Resource.asm, Resource.inc, Makefile and a "test.h" file.

22.5.3 Add File and Resource to Project

1. Click [Project]→[Add to Project]→[Files] and [Resource] to display Add Files dialog box. You can also right click on Workspace window and select "Add Files to Project..." to toggle on the dialog box.



2. Select one or more files in file list box.

3. Click [Open].

Open Project

When open an existing project, µ'nSP® IDE will resume the last setting. Three ways are provided to open project.

4. Click [File]→[Open Project] to display "Open" dialog box.
5. Select 'Project Files(*.spj)/*(.scs)/Pack File(*.pak)' in type list box.
6. Select 'Project Files(*.spj)/*(.scs)/Pack File(*.pak)' in type list box.

The second way to open a project is use [File]→[Recent Projects]. Recent Projects menu lists at most 4 recent accessed projects.

The third way:

1. Click [File]→[Open] to display "Open" dialog box.
2. Select 'Project Files(*.spj)' in type list box.
3. Select a project in project list box and select open type as text. Then, click [Open] or double click the project directly to open it.

Note: Check box "Preview" can be used to preview the text file in Open dialog box.

22.5.4 Save Project

1. Click [File]→[Save Project] to save the project file and the configuration of current project.

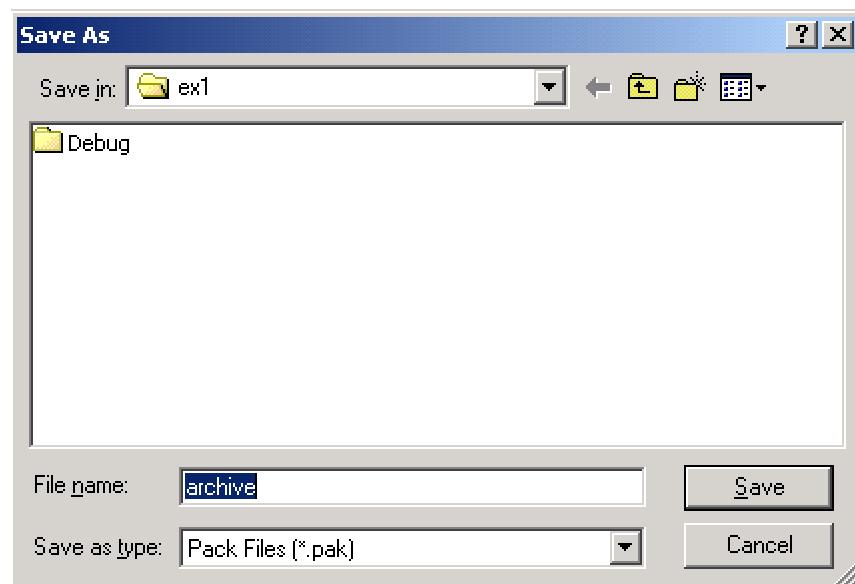
2. After you rebuild and close one project, system can auto save project.

Pack Project

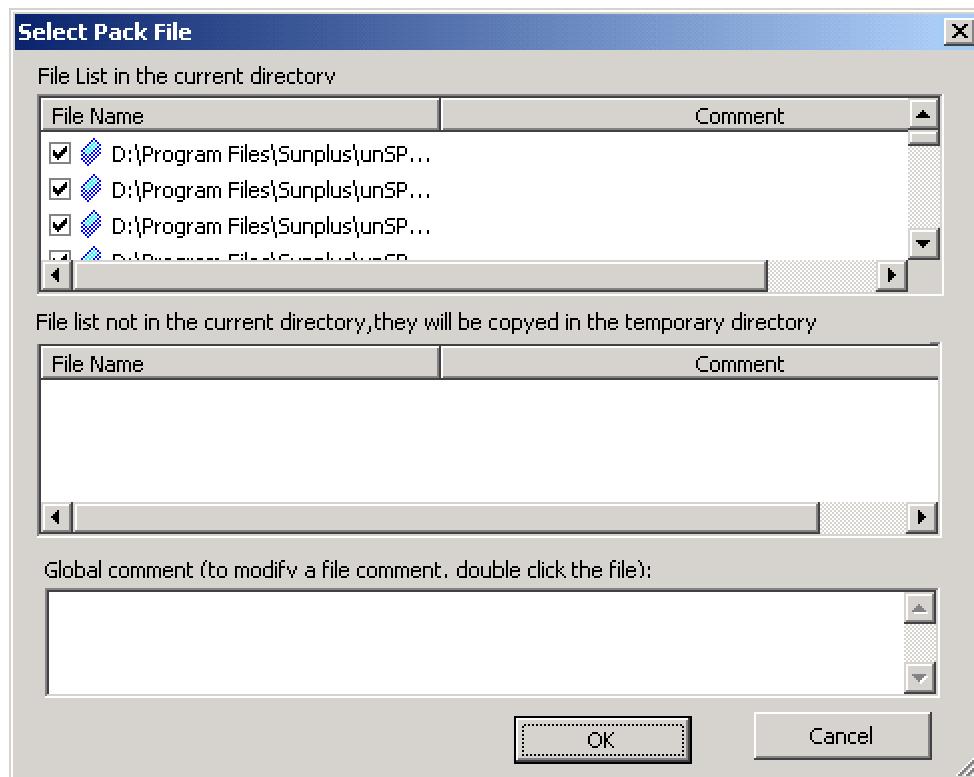
Pack project function can help you to pack all information about current project to one .pak file, anytime you extract the pack file, you can use the project with the same configuration.

3. Create Pack File

(1) Click [File]→[Pack Project], the Save As dialog box will be pop-up.



(2) Input the pack filename in Filename box and click [Save]. After you click Save button, the Select Pack File dialog box is shown.



You can select which file to be packed to file in File List in the current directory.

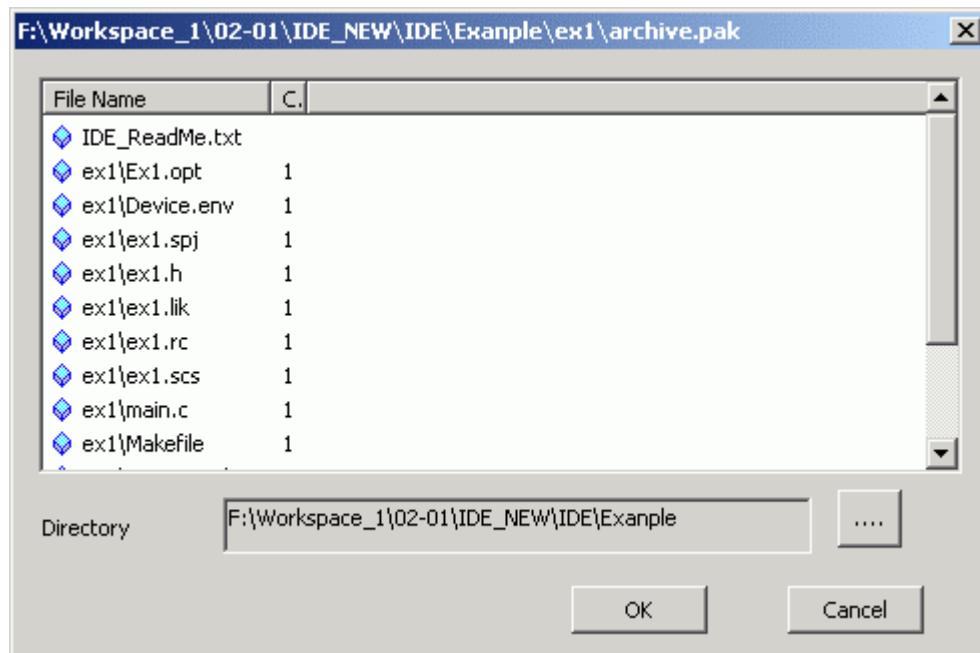
The file you have added from other directory will be copied in the temporary directory.

In the Global comment box, you can specify comment for all files of current project. To specify the comment for single file, you should double click on the file name on the upper two windows.

- (3) Click [OK]. The .pak file will be generated.

4. Extract Pack File

- (1) Click [File]→[Open Project] to open the .pak file. And the files in the project will be shown.



- (2) Set the directory to extract .pak file.

22.5.5 Create Object

An internal object is a binary output file of current project. The µ'nSP® IDE creates an object according to source files and the project description. Generally, the µ'nSP® IDE follows the following criteria to create an object:

1. The type of binary output file (format: S37, TSK).
2. The setting of compiler and linker.

You can select Debug and Release from pull-down list on build toolbar to create a new debug object/release object at any time.

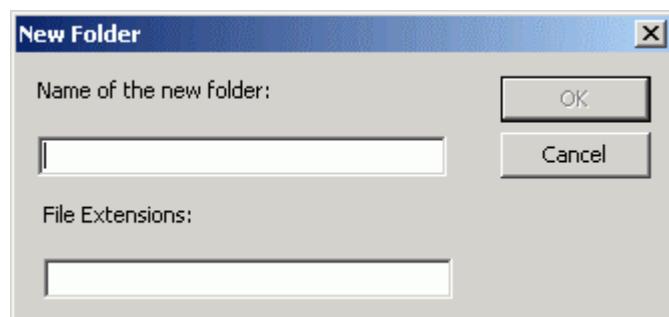
22.5.6 Use Group in Project

After creating a new project, µ'nSP® IDE will automatically create three groups (Source Files, Head Files and External Dependencies) to the project. Certainly, you can also create additional groups to save files if necessary. Groups can display the logic relationship of files. For example, you can create a group that contains all user interface files or other files such as project description, document record, test group, etc. All these files are shown with different icons.

After creating a new project, µ'nSP® IDE will automatically create three groups (Source Files, Head Files and External Dependencies) to the project. Certainly, you can also create additional groups to save files if necessary. Groups can display the logic relationship of files. For example, you can create a group that contains all user interface files or other files such as project description, document record, test group,etc. All these files are shown with different icons.

22.5.7 Add Group to Project

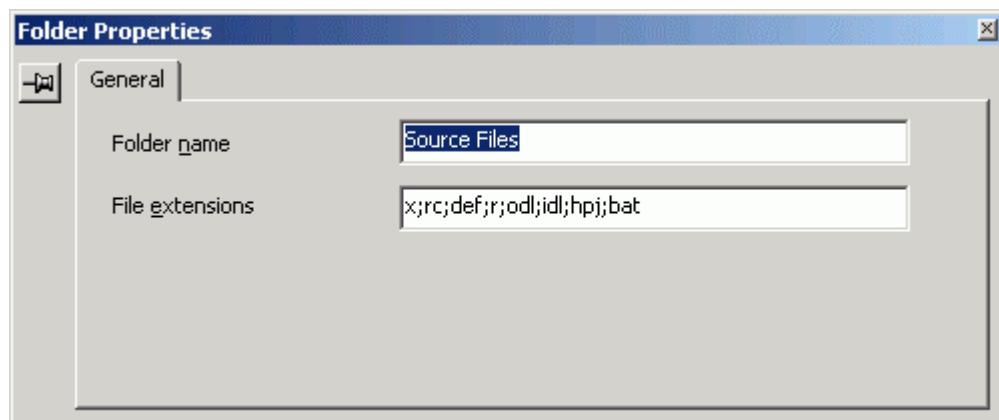
1. In project window, point the cursor on a project or a group and then click on the right button of a mouse to display hot key menu.
2. Select "New Folder" on the menu.
3. A New Folder window as below will pop-up, input the new group's name in Name of the new folder box.



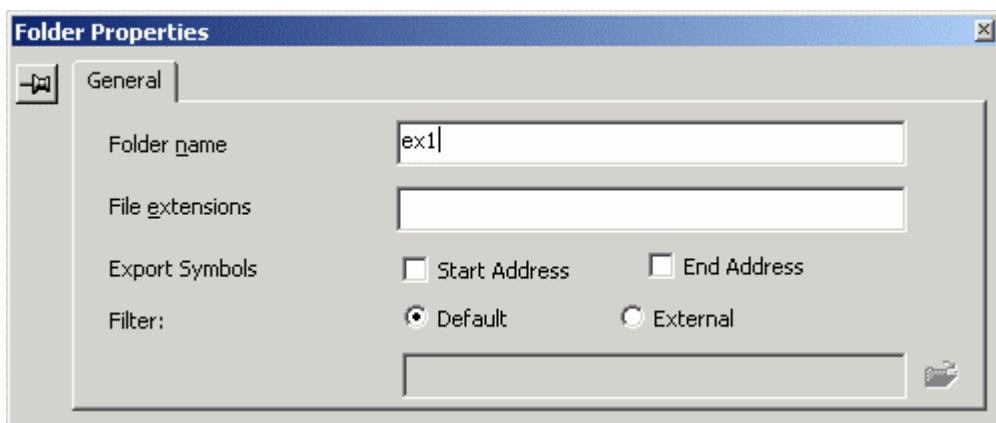
4. Key in the folder name in the upper box and specify the file type of current folder in File Extensions box.
5. The folder settings will be apply to the resource files which added or moved to this folder.

22.5.8 Change Group Property

To change the group property in FileView window, you should select group in FileView window and right click to select "Properties".



To change the group property in ResourceView window, you should select group in ResourceView window and right click to select "Properties".



Folder name and File extensions can specify folder name and the file type of current folder. Export Symbols and Filter can change the selected folder's export symbols and select the corresponding external filter. Before linking, you can choose the external filter to convert the resource to SUNPLUS format.

The external filter should follow the format below:

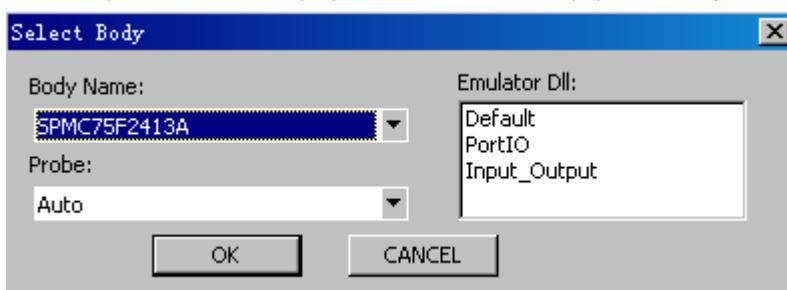
Filter.exe <input file> <*output.sp> *output file extension must be ".sp"

22.5.9 Move File and Group

1. Select one or more file and group in project window.
2. Drag the selected file and group to project level and destination group.
3. If you are moving files, the destination group settings will be apply to these files.

22.5.10 Select Body

Click [Project]→[Select Body], you can select the body type for programming.



Body Name: Select body.

Probe: Select probe type which is connected with ICE.

Emulator Dll: Select emulator DLL.

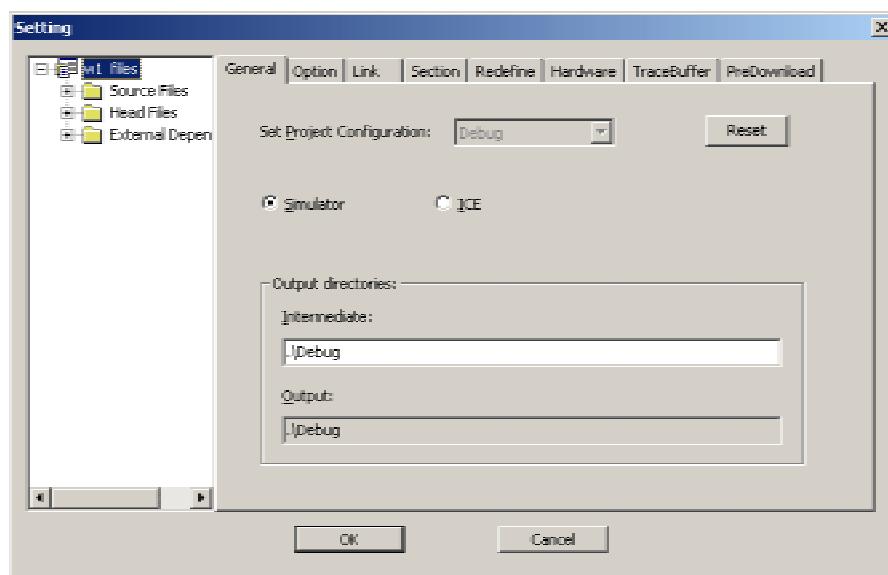
22.5.11 Set Option for Project

For most projects, options are set at project level. You can set options for different objects that are created by system/user. Generally speaking, µ'nSP® IDE will create two types of objects: Debug and Release. Project can display the structure of options for every object. The options given for the project level also apply to all files in the project.

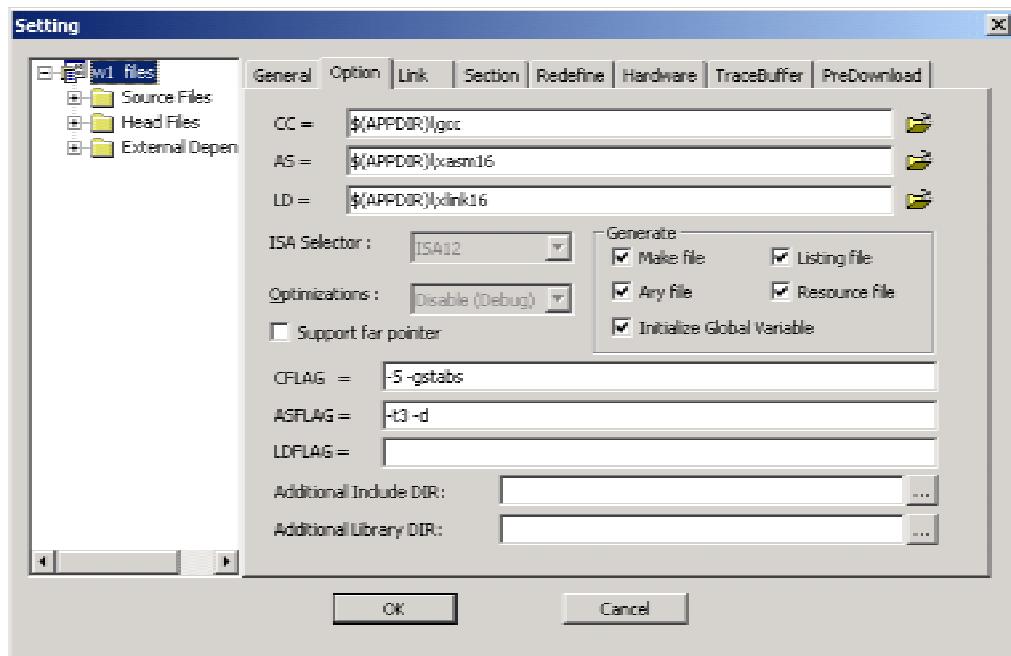
Set Option for Project

1. Open a project or create a project.
2. Click [Project]→[Setting] to display "Setting" dialog box.
3. Click on each label for further setting.

[General]:



1. Set Project Configuration: Display the version for the project to be used.
2. Simulator/ICE: Select µ'nSP® IDE running mode. If ICE mode is selected, you should connect an emulation board to your computer through a parallel port. If simulator is chosen, all data will be stored into buffer.
3. Intermediate: Specify directory for intermediate files. Intermediate files normally are generated during compilation.
4. Output: Display the terminal file's directory. Normally, it is the same as the intermediate directory.
5. Reset: Reset default configuration.

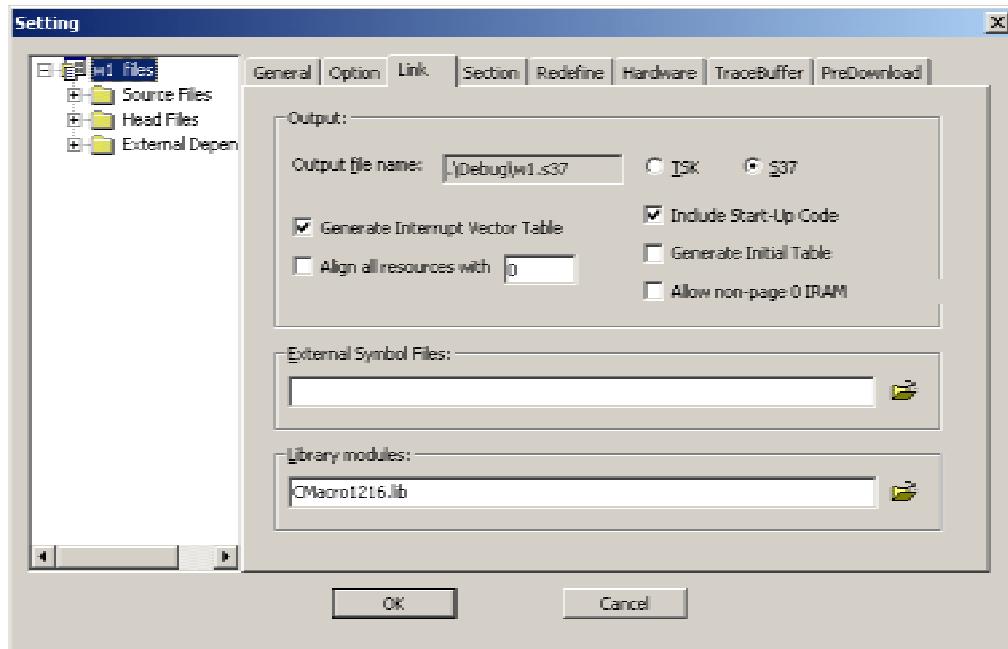
[Option]:


1. CC: Specify the C Compiler location.
 2. AS: Specify the Assembler location.
 3. LD: Specify the linker location.
 4. CFLAG: Specify the C Compiler FLAGS.
 5. ASFLAG: Specify the assembler FLAGS.
 6. LDFLAG: Specify the linker FLAGS.
 7. Optimizations: Select the Optimization Type you want, the optimization flags will be changed automatically.
 8. ISA Selector: Display instruction set.
 9. Support far pointer: Check if IDE supports far pointer.
- Note:** If “Support far pointer” is checked, “CC” will be specified as “xgcc”, and will link “Cmacro1x32.lib”; otherwise, “CC” will be specified as “gcc”, and will link “Cmacro1x16.lib”
10. Makefile: Check if auto updating the makefile.
 11. Listing file: Check if generate the listing file.
 12. Ary file: Check if auto updating the array file.
 13. Resource file: Check if generate the resource file(resource.inc and resource.asm).
 14. Initialize Global Variable: Check if initialize the global variable.
 15. Additional Include Dir: Set include file directory.
 16. Additional Library Dir: Set library file directory.

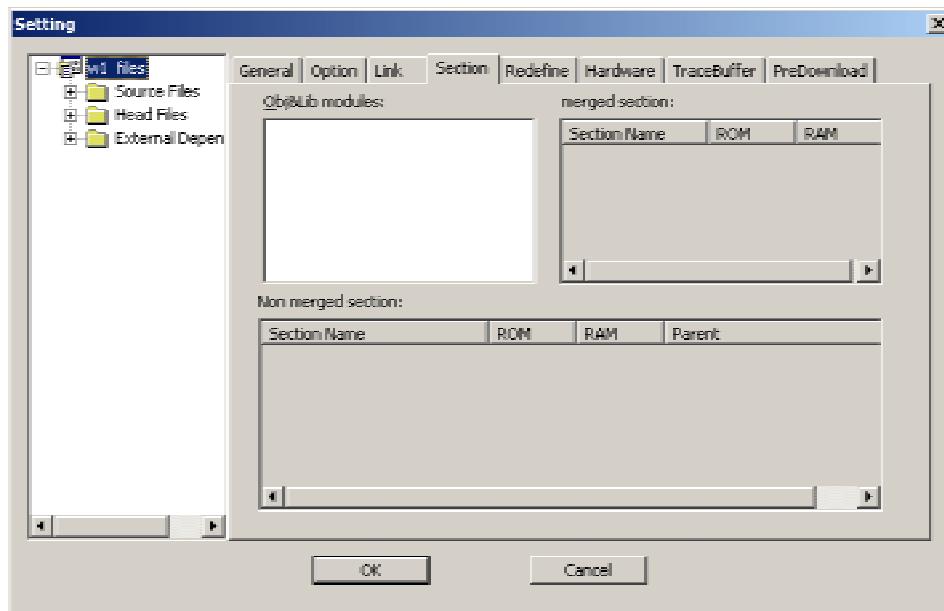
The CFLAG, ASFLAG, LDFLAG can be specified with different parameters in Debug and Release mode.

While changing the build mode in [General], the parameters will automatically switch to proper ones.

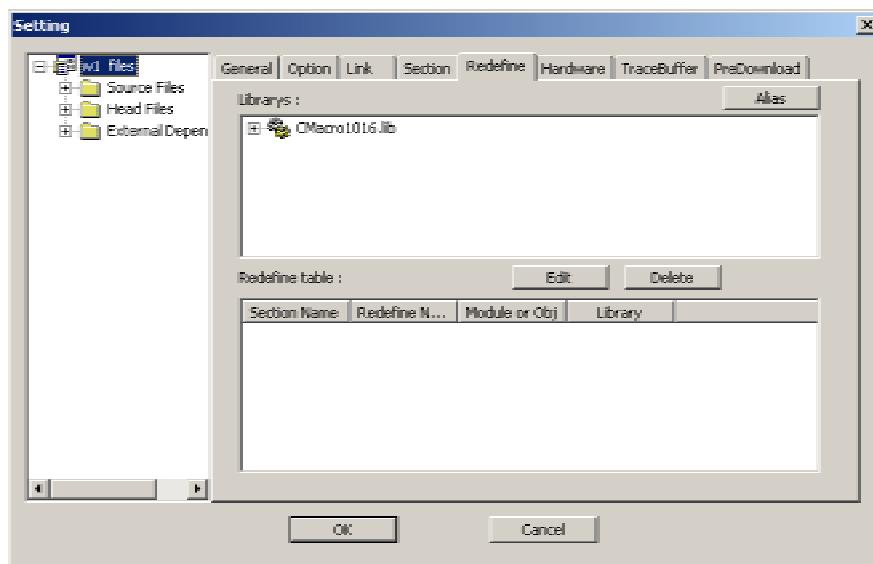
You can set “-tsklen xxxx” in “LDFLAG to limit output TSK length. For example, if you want to limit tsk length to 0x6ffff, you can set “-tsklen 6ffff”.

[Link]:

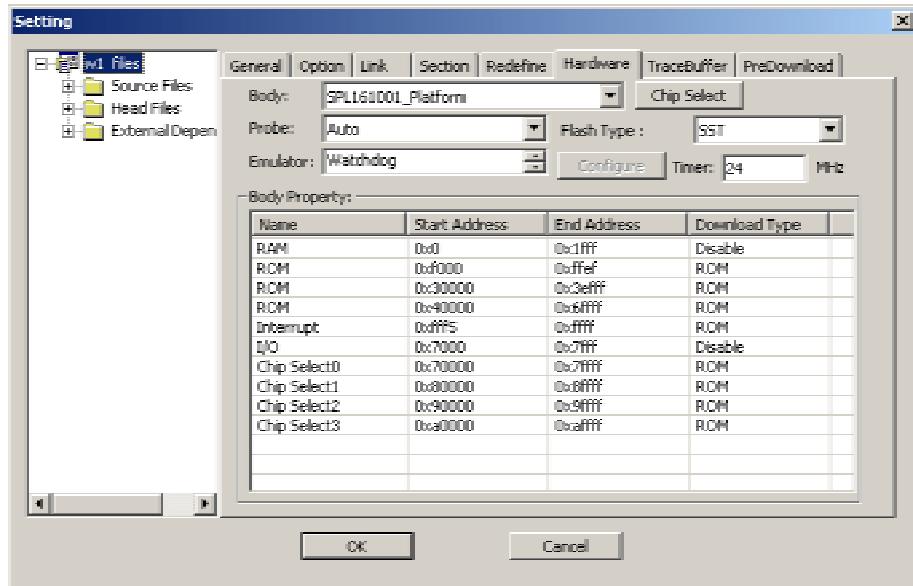
1. Output file name: Specify the output file's name.
2. TSK/S37: Types of object file (binary/ASCII (Motorola S37)). Before you select object file type, you should select both Makefile and Ary file at Option page.
3. Generate Interrupt Vector Table: Uncheck it, if you don't want to link produce the Interrupt Vector Table in the project output.
4. Include Start-Up Code : Uncheck it, if you don't want to link produce the default start-up code in the project output.
5. Align all resource with : Check it and input the align base, if you want to align all resource with a specify align base.
6. Generate Initial Table: Uncheck it if you don't need an initial table in the project output.
7. External Symbol Files: Input the other symbol files (*.sym) needed for reference link in the current project.
8. Allow non-page 0 IRAM: Enable the IRAM to occupy the non page 0.
9. Library modules: Specify and show all library-modules included in current project.

[Section]:


1. Obj & Lib modules: Show all object and library modules in current project.
2. Emerge section: List the emerged segments in current project.
3. Non-emerged section: List the non-emerged sections in current project. You can change the address or align base of these sections by double click on the ROM field, and the specified sections will be located at proper addresses with align base after re-linking this project.

[Redefine]:


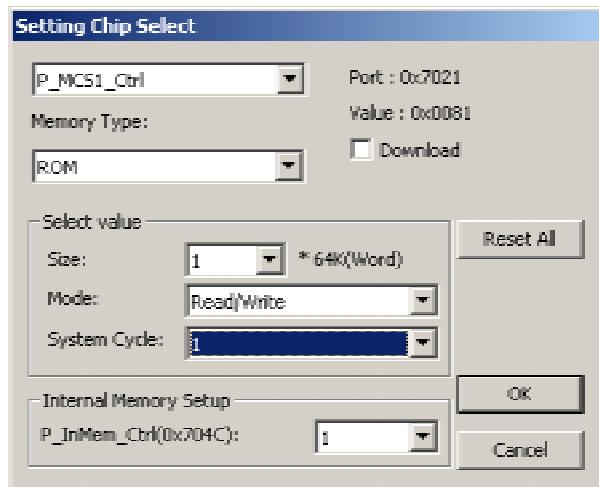
1. Alias: Select an section from the list window of libraries and redefine it as another name
2. Edit: Edit the selected item in the Redefine table list window. Equal to double click on the item in the Redefine table list.
3. Delete: Delete a selected item in the Redefine table list window.

[Hardware]:


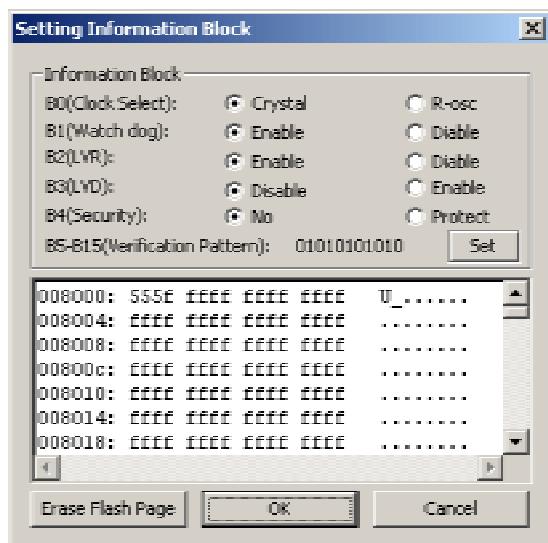
For the body with Chip Select and Information Block, Chip Select button and Information Block button can be provided.

1. Body: Select a body. The linker and simulator are based on the body description to link and simulate.
2. Probe: Select probe type which is connected with ICE.
3. Emulator: Select the external device emulator for selected IC, the emulator is a dll specified at cpt file of correspond body.
4. Timer: Set emulate clock frequency for Simulator and Emulator.
5. Configure: Set emulator.
6. Chip Select: Set chip select.
7. Body property: Show the memory mapping.
8. Flash Type: Select flash type, after you select a body supporting flash.

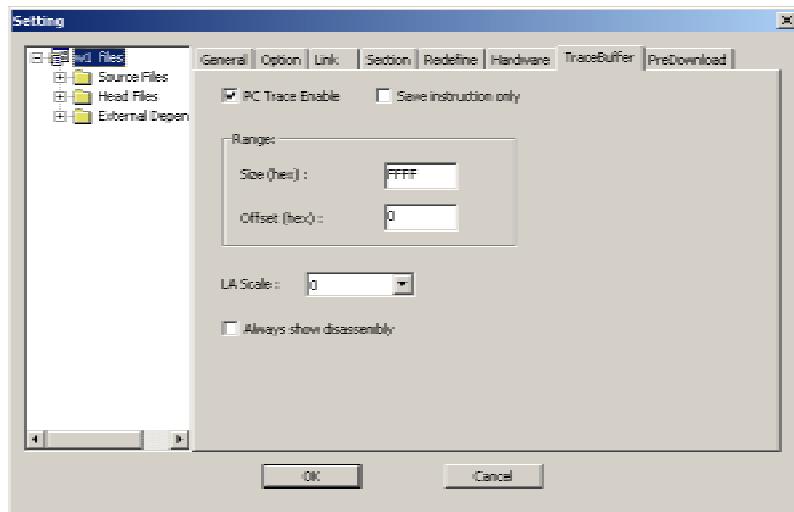
Set chip select:



Set information block:

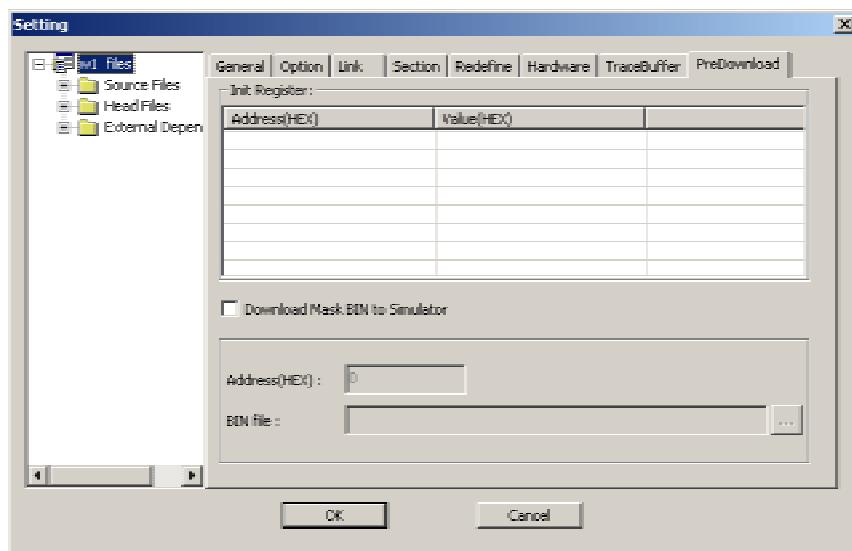


[TraceBuffer]:



1. PC Trace Enable: Enable PC Trace function(please make sure your ice board has PC Trace capability when you are in ICE mode).
 2. Save Instruction Only: If checked only instruction fetch will be saved in pc trace buffer, else will save all memory read/write in pc trace buffer.
 3. LA Scale: Configure the LA scale.
 4. Size & Offset: Set the size and offset of the buffer.
 5. Always show disassembly: Check to make the disassembly window always display on the main interface.

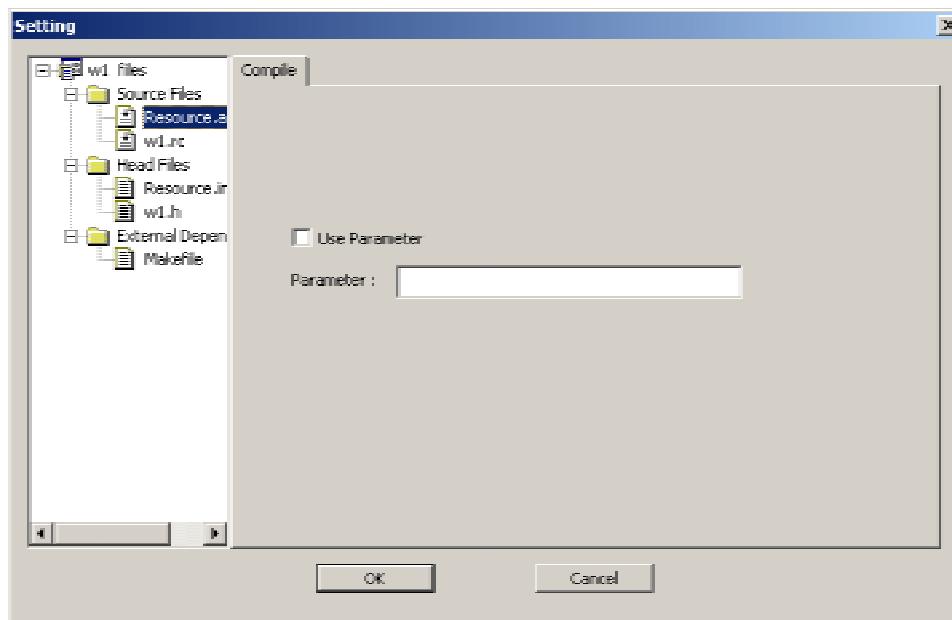
[PreDownload]:



1. Init Register: In ICE mode, write data to memory before download.
 2. Download Mask BIN to Simulator: In simulator mode, write data to the memory before download. It can begin from 'Address' and the data is got from 'BIN file'.

Set Option for File

You can specify parameter for compiler. To do this, click [Project]→[Setting...] to enable the Setting dialog box, then select a file on the directory tree from the left side, a Compile page will be displayed.



Check "Use Parameter" and then input the parameter in the text box. Valid parameters are listed here.

- S: Compile to assembly language
- E: Run only the preprocessor on the named C programs
- o file: Place output in file 'file'
- help: Print a description of the command line options recognized by gcc
- ansi: Support all ANSI standard C programs. Turn off certain features of GCC that are incompatible with ANSI C. The '-ansi' option does not cause non-ANSI programs to be rejected gratuitously. For that, '-pedantic' is required in addition to '-ansi'.
- pedantic: Issue all the warnings demanded by strict ANSI C and ISO C++
- w: Inhibit all warning messages
- Wall: Enable all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros.
- Werror: Make all warnings into errors
- Q: Make the compiler print out each function name as it is compiled, and print some statistics about each pass when it _nishes.
- Dmacro: Define macro macro with the string '1' as its definition. -Dmacro=defn: Define macro macro as 'defn'. All instances of '-D' on the command line are processed before any '-U' options.
- Umacro: Undefine macro macro
- gstab: Produce debugging information for debuggers
- Idir: Add the directory 'dir' to the head of the list of directories to be searched for header files.
- O0: No optimization
- O1: The compiler tries to reduce code size and execution time.

- O2: Optimize more than O1. Nearly all supported optimizations that do not involve a space-speed tradeoff are performed.
- O3: Optimize more than O2. This turns on all optimizations -O2 does. In addition, turn function in-lining on.
- Os: Optimize for size. This enables all -O2 optimizations that do not typically increase code size. It also performs further optimizations designed to reduce code size.

22.5.12 Build Project

Use µ'nSP® IDE to build and rebuild the programs and libraries defined in the project. To build a project, µ'nSP® IDE can only process the files which have been edited after the last compiling. To rebuild whole project, µ'nSP® IDE will rebuild current project and all project dependencies.

In general, µ'nSP® IDE set default option for the new object while creating a project. Debug object contains adequate symbol debug information, which is used by µ'nSP® IDE debugger, and all optimized options will be invalid in debug because those options will hamper the debugging. Release object does not have symbol debug information and all optimized options are valid in creating an object.

Every object has the specified directory for intermediate file and output file.

Build Current Object

Be sure you register µ'nSP® IDE from [Help]→[About SUNPLUSIDE]. After registration, you are able to compile, link and debug the project.

1. Click [Build]→[Compile] to compile the current active file in edit window. (Equal to right click on *.asm, *.c, etc. on Workspace window and select Compile)
2. Click [Build]→[Build] to build the object. (Equal to right click on project name on Workspace window and select Build)
3. Click [Build]→[Rebuild All] to rebuild the project and all project dependencies.
4. Click [Build]→[Stop Build] to stop building.

Build information is shown in Build page of output window. Compiling is successful if no error existed. While compiling, you can also use µ'nSP® IDE to complete other operations, but some menu commands and toolbar buttons may be invalid. Before executing Stop Build, µ'nSP® IDE will try to close the current command first. If current command is unable to be stopped, the system will wait until the current job is completely executed.

22.5.13 Run Program

After you build the project, you can startup the application program, or run the application program and DLL (Down-Line Load) in integrated debugger.

Click [Build]→[Start Debug]→[Download] to download an executable program, then, use other commands in the Debug menu to run and debug it.

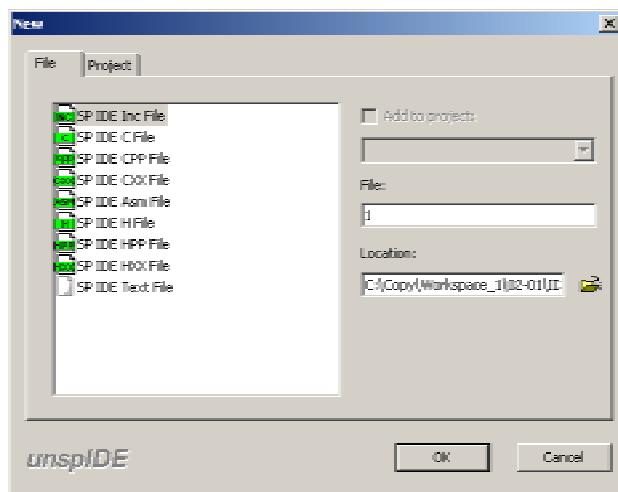
22.5.14 Load Program

You can click [File]→[Load Program] to load executable file(*.bin, *.TSK, *.S37) and debug/run.

22.6 Source Code

22.6.1 Create Document

Inc(include file), Asm(assembly file), C, CPP, CXX, H, HPP, HXX, Text are supported in this tool. Click [File]→[New] to open "New" dialog box.



1. Click File label.
2. Select the file type in the list box. To add a new file to a project, select Add to project and project name.
3. Input the new file name in File box.
4. Specify the file's directory in Location box.
5. Click [OK].

Open Document

Click [File]→[Open] to show Open dialog box. The open type can be selected as binary/text.

Another way to open is:

Click [File]→[Recent Files] to select a document and open it directly. Recent Files menu lists at most 8 recent accessed files.

22.6.2 Save Document

A document name on title bar or from Window menu marked with "*" means the current file being edited.

If close the document with "*", editor will prompt you to save it. Once you save the document, "*" will disappear.

1. Click Save on toolbar / [File]→[Save] to save file with the same filename.
2. Click [File]→[Save As] to save file with different filename.
3. Click [File]→[Save All] to save all open documents.

22.6.3 Bookmark

You can insert a bookmark in your document on text editor. However, the bookmark will be gone at the time of document closed.

1. Set Bookmark

- (1) Move the cursor to a destination where to insert a bookmark, click [Edit]→[Bookmark] to add bookmark. You can use Ctrl+n(n=0~9) to set bookmark with number.
- (2) Click [Edit]→[Previous Bookmark] to move the cursor up to the previous bookmark.
- (3) Click [Edit]→[Next Bookmark] to move the cursor down to the next bookmark.

2. Delete Bookmark

- (1) Move the cursor to the line containing a bookmark, click [Edit]→[Bookmark] to delete the bookmark.
- (2) Click [Edit]→[Clear All Bookmark] to delete all bookmarks.
- (3) Close the document to delete all bookmarks.

22.6.4 Find Text

Text editor provides three ways to find a text:

1. Find Text in an Opening Document

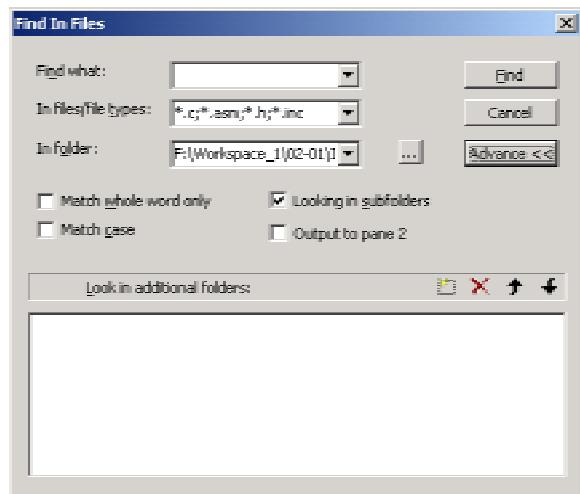
- (1) Click [Edit]→[Find] to open "Find" dialog box.
- (2) Input the string to be found in "Find What" box.
- (3) Select "Match Case"/"Match Whole Word Only" to find the string matched with case/ word.
- (4) Select "Up"/"Down" to search for the string from the current location of the cursor to the beginning of the document or to the end of the document.
- (5) Click [Mark All] to mark all matched strings with bookmarks, and click [Unmark All] to unmark all matched strings with bookmarks.
- (6) Click [Find Next] to find matched string.
- (7) Click [Edit]→[Find Next] and [Find Previous] to find the next and previous matched string.

2. Replace Text

- (1) Click [Edit]→[Replace] to open "Replace" dialog box.
- (2) Input string A to be found in "Find What" box. Input string B to replace the found text in "Replace With" box, if leaving blank in the box, editor will delete all found string.
- (3) Click check box "Match whole word only" and "Match case" to find the string matched with whole word/case.
- (4) Selection/Whole file: Replace the string in current selection only/ entire file. Selection option will be disabled when no text is selected.
- (5) Click [Find Next] to ignore the current matched string. Click [Replace]/[Replace All] to replace the current matched/all matched string in document.

3. Search for Text in Disk File

- (1) Click [Edit]→[Find In Files] to show "Find in Files" dialog box.



- (2) Input the string in "Find What" box.
- (3) Select file type in In files/file types pull-down list.
- (4) Input the directory in "In folder" box or click button "..." to browse path.
- (5) Click "Match whole word only"/"Match case" to select the string matched with whole word or case sensitive. Select "Looking in subfolders" to search in subfolder (Default option). Select "Output to pane 2" to output the finding result to page Find in Files2 of output window. Or the result will output to Page Find in Files1.
- (6) Click [Advance>>] to set directory for finding.
- (7) Click [Find].

22.6.5 Binary Editor

1. Open Binary File

- (1) Click [File]→[Open] to pop up "Open" dialog box. Equal to click [File]→[Recent Files].
- (2) Select "Binary" in Open as box.
- (3) Select the filename in file list box.
- (4) Click [OK].

Note:

If another editor is editing the resource, you should close that editor before using binary editor.

2. Edit Binary Data

- (1) Click in HEX section or ASCII section of binary editor.
Keys [Up], [Down], [Left], [Right], [Page Up], [Page Down], [Home], [End] are effective in binary editor.
- (2) Modify the content in a code section.
- (3) Save the modification.

22.7 Resource

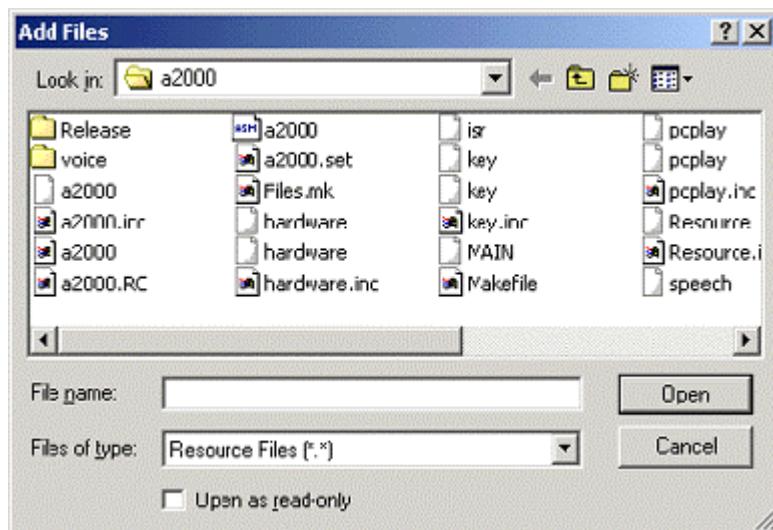
22.7.1 Resource ID

Resource ID contains a series of characters. After a resource is added, the default name of RES_* (where * is the filename) and resource ID are assigned.

Edit Resource

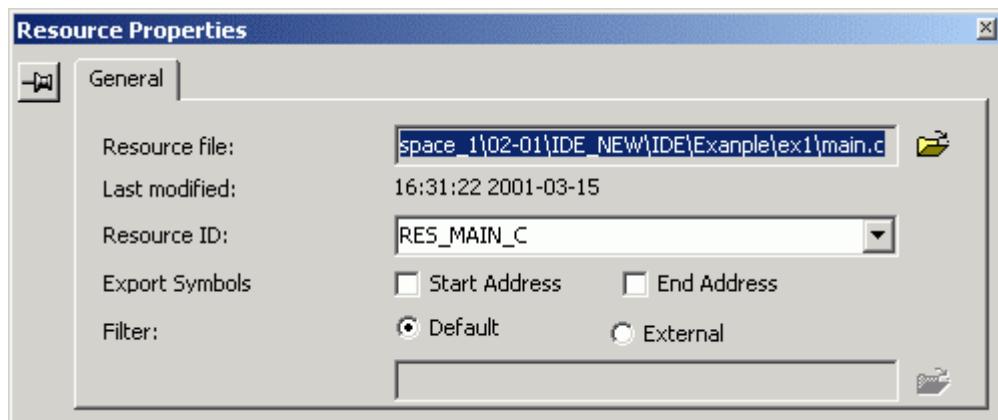
Add Resource

On the resource page within Workspace window, select one group, right click, and then show a hot key menu. Click [Add Files to Folder] to add a new resource. The "Add Files" dialog box is shown as follows:



22.7.2 Change Resource

On resource page within Workspace window, select a resource and right click, a hot key menu shown as follows:



[Open]: Open the selected resource file in binary edit window.

[External Editor]: Use the external editor to open the resource file.

[Remove]: Remove the selected resource file from the current project.

[Hide]: Hide the Workspace window.

[Properties]: Change the selected resource file's location, resource ID, export symbols and select the corresponding external filter. Before linking, you can choose the external filter to convert the resource to

22.7.3 SUNPLUS format

The external filter should follow the format below:

Filter.exe <input file> <*output.sp> *output file extension must be ".sp"

22.8 Debugger

The integrated debugger helps users to locate program's logic errors after the syntax errors are cleared.

The µ'nSP® IDE is able to monitor where the program has been executed or stop program at a specific point for further evaluation, etc. All contents in variables, memory and registers can be shown on your screen in debug mode.

22.8.1 Run Control

First, click [Build]→[Start Debug]→[Download] to download a program and start debugging program in debug mode.

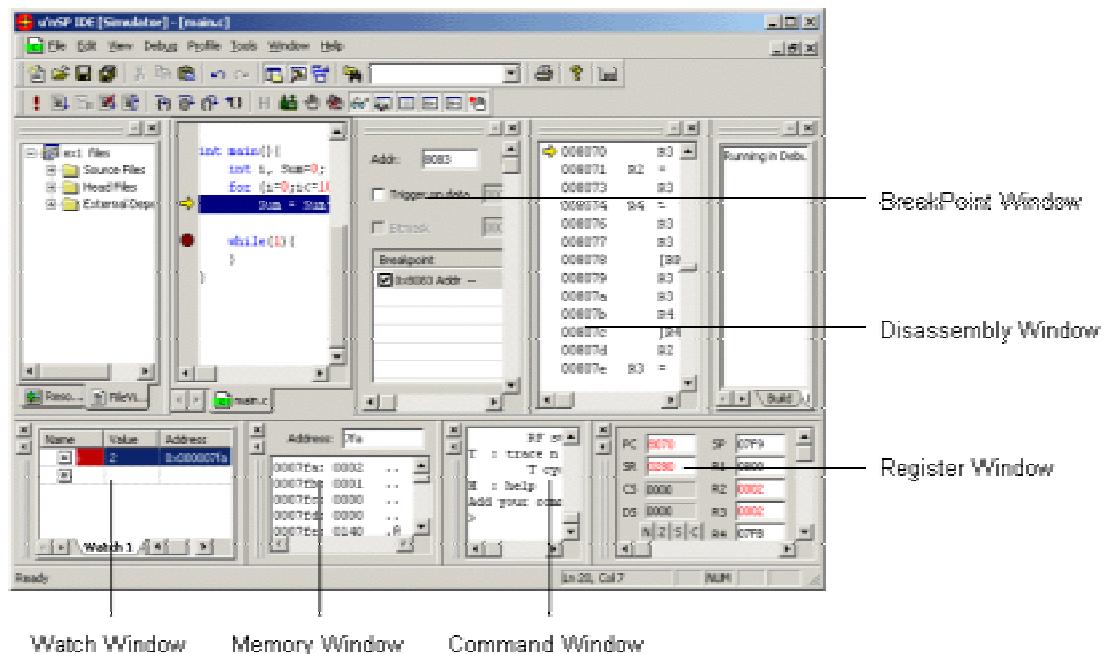
Once the program is being debugged, you can use menu commands on Debug menu to control program flowing. The following table lists all commands and descriptions.

Menu	Function
Go (F5)	Run program from current position to the first breakpoint or to the end of program.
Restart (Ctrl+Shift+F5)	Restart running program.
Stop Debug (Shift+F5)	Stop debugging the program and return to normal edit state. System will automatically save all breakpoints and resume them for the next run.
Break	Break running.
Step Into (F11)	Step into the next called function (step by step run the program).
Step Over (F10)	Step over the next called function and continue to execute the followed instructions.
Step Out (Shift+F11)	Return to the calling program from the called function and continue to run the next command. By using Step into and Step Out, you can trace part of the source code.
Run to Cursor (Ctrl+F10)	Run from current command line to the cursor's line. It is the same as set one temporary breakpoint at the cursor's command line.

You can step over the specified code and continue to debug the rest of program. You can also run again to test the program after changing variable's content. Furthermore, system permits to run the whole program or part of it by single step running.

22.8.2 Debug window

In debug mode, Debug menu will appear on the main menu bar. System uses a series of windows to display debug information. You can use the commands on View menu to visit these windows.



Memory Window:

Display the contents of memory.

Register Window:

Display current contents of the general registers and CPU status register.

Command Window:

Display the command window.

BreakPoint Window:

Display the contents of breakpoints.

Watch Window:

Display the content of variables and expressions. Input and edit expression allowed.

Disassembly Window:

Display the disassembly contents of memory.

History Buffer Window:

Display the executed instructions, status and memory contents

22.9 Profile

22.9.1 Profile Function

Profile is a forcible analytical tool in µ'nSP® IDE. It can be used when program is confirmed. Profile functions are as follows:

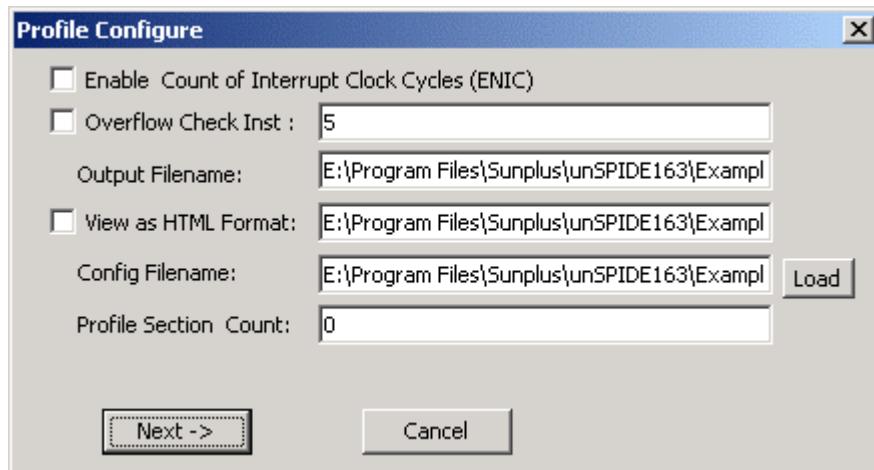
1. Provide information for the best codes utilization.
2. Detect the running program, to measure whether the arithmetic is effective.
3. Check whether certain code segment is over written by software test program.

Furthermore, µ'nSP® IDE provides profile function that allows you to analyze part of program. The analysis includes instruction cycles, IRQ, label flow analysis and some significant information to assist improving program performance. Before running profile, you must stop executing and debugging program.

To stop debugging mode, select [Debug]→[Stop debug].

22.9.2 Profile Operation

1. Click [Build]→[Profile], you will see a dialog box as follows:

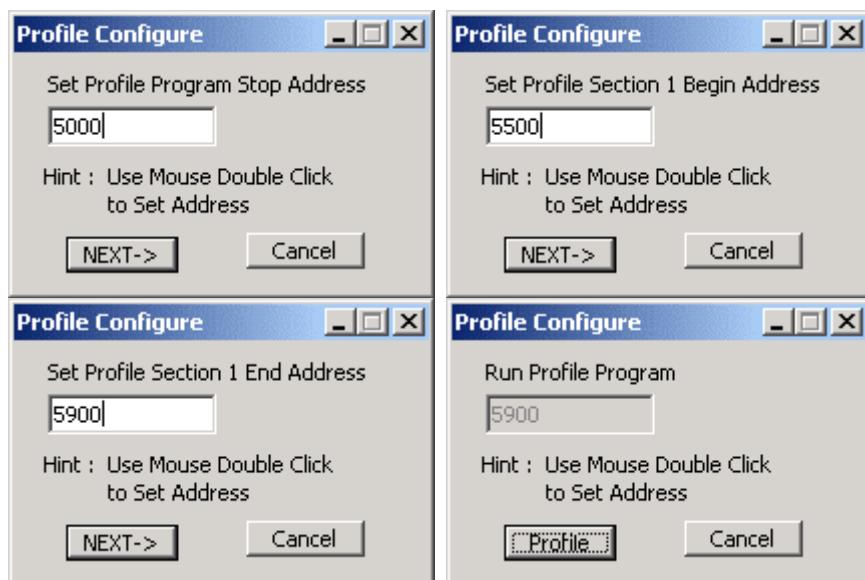


- (1) Enable Count of Interrupt Clock Cycles (ENIC): The ENIC is for continuously analyzing in IRQ. If the profile section (a program section being analyzed) is located within the IRQ routine, you must enable this option.
- (2) Overflow Check Inst: Set the number of instruction for warning when the overflow flag is not checked.
- (3) Output Filename: Final filename.

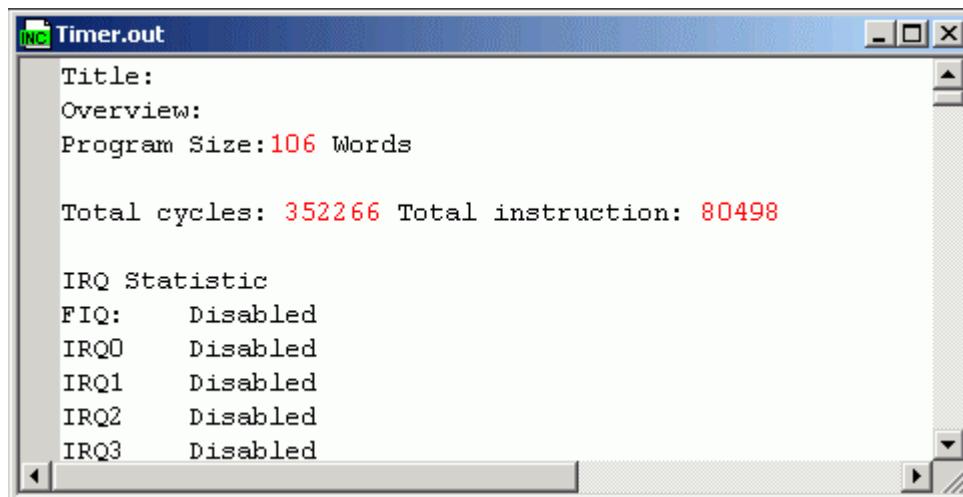
- (4) Config Filename: The configuration can be saved to a file and reloaded after.
- (5) Profile section count: Number of section to be analyzed. N is not greater than 32. A profile section is the section of one program to be analyzed.

2. Click [Next] in profile configure box for additional dialog box.

In this dialog box, you should specify the end address for the analyzed section. You can either enter a specified address into the box or double-click on the instruction line. Also, you need to specify the end address and start address for each profile section.



3. Click [Profile] to make a profile analysis. Generate a profile may be slow due to the size and path of section executed. You can still see a profile file on the screen. That is the result at the time you run the profile through the time you stop it.

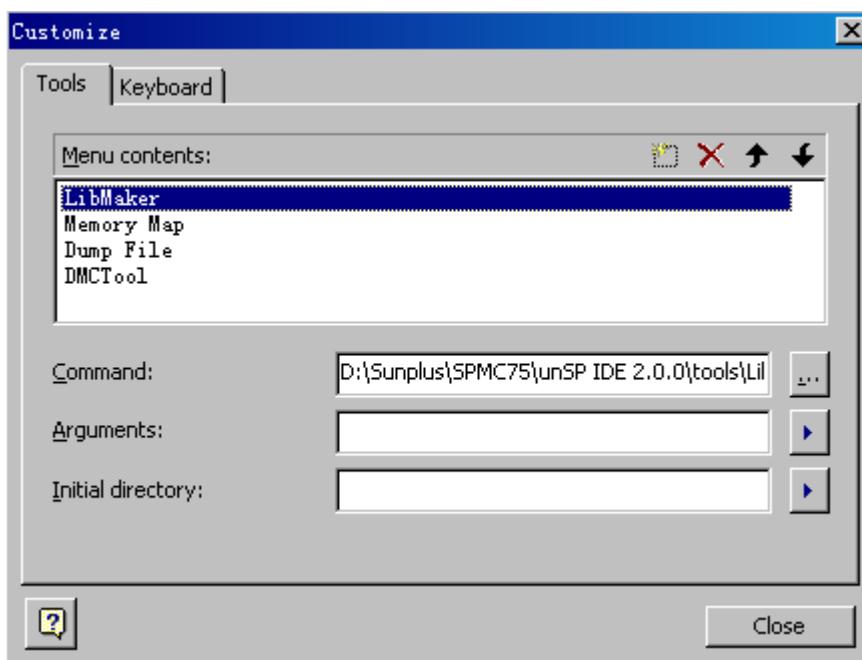


22.10 Tool

22.10.1 Customize Developer Studio

The more applications you work, the more you would like to customize the µ'nSP® IDE to a comfortable and efficient environment for you. The µ'nSP® IDE actually provides a direct way to accomplish it. For example, you can set hot key and external application program to be called in µ'nSP® IDE. Once setup is completed, system will save and memorize it.

22.10.2 Tool



This view provides a way to add external application. Then, the user-defined tool will be enabled on Tools menu, and you can run it within current integrated development environment. Simply click the application on the Tools menu, µ'nSP® IDE automatically calls the relevant application.

You can add up commands to the Tools menu. These commands can be associated with any program that will run on your operating system. You can also specify arguments for any command that you add to the Tools menu.

1. Add Command to the Tools Menu

- (1) To add a tool, in the Menu Contents box, Click  to add a new menu content. You can also double-click the blank line of the list (indicated by an empty rectangle), type the name of the tool as you want it to appear on the Tools menu, and press [ENTER].

Highlight the name of the tool you just entered.

- (2) In the Command box browse or type or select the path and name of the program, for example, C:\WINDOWS\NOTEPAD.EXE.
- (3) In the Arguments text box, browse or type any arguments to be passed to the program. Click the drop-down arrow next to the Arguments text box to display a menu of arguments. Select an argument from the list to insert argument syntax into the Arguments text box.
- (4) In the Initial Directory box, type the file directory where the command is located. Click the drop-down arrow next to the Initial Directory text box to display a menu of directories. Select a directory from the list to insert directory syntax into the Initial Directory text box.

You can change the default menu name of the newly added tool by editing the Menu Text box.

Check the argument description from Specify Argument for Tool.

2. Specify Argument for Tool

You can use argument macros to specify arguments for Tools menu command. The argument macros shown in the following table.

Context Menu	Argument Macro	Description
File Path	\$(FilePath)	The complete filename of the current source (defined as drive+path+filename); blank if a nonsource window is active.
File Directory	\$(FileDir)	The directory of the current source (defined as drive+path); blank if a nonsource window is active.
File Name	\$(FileName)	The filename of the current source (defined as filename); blank if a nonsource window is active.
File Extension	\$(FileExt)	The filename extension of the current source.
Current Line	\$(CurLine)	The current cursor line position within the active window.
Current Column	\$(CurCol)	The current cursor column position within the active window.
Current Text	\$(CurText)	The current text (the word under the current cursor position, or a single-line selection, if there is one).
Current Directory	\$(CurDir)	The current working directory (defined as drive+path).
Target Path	\$(TargetPath)	The complete filename of the current target (defined as drive+path+filename).

Context Menu	Argument Macro	Description
Target Directory	<code>\$(TargetDir)</code>	The directory of the current target (defined as drive+path).
Target Name	<code>\$(TargetName)</code>	The filename of the current target (defined as filename).
Target Extension	<code>\$(TargetExt)</code>	The filename extension of the current target.
Workspace Directory	<code>\$(WkspDir)</code>	The directory of the current workspace (defined as drive+path); blank if no workspace is currently open.
Workspace Name	<code>\$(WkspName)</code>	The current workspace name (defined as filename); blank if no workspace is currently open.
Project Directory	<code>\$(ProjectDir)</code>	The directory of the current project (defined as drive+path).
Current Body	<code>\$(CurrentBody)</code>	Current body type.

3. Edit Command on the Tool Menu

In the Menu Contents box, select the command that you want to edit.

You can change the tool position in the list and delete tool.

To move the command up one position in the menu, choose the Move Up button.

To move the command down one position in the menu, choose the Move Down button.

To delete a tool, choose the Delete button.

To change the menu text, command line, command-line arguments, or the initial directory, type the new information in the appropriate text box.

22.10.3 Keyboard



In this window, you can set hotkey to toggle on certain command with keyboard. You can delete or change key assignments, and you can assign hotkey for each editor. You can also reset all hotkeys to their default setting.

1. Assign and Delete Shortcut Key

In the Category list, select the menu that contains the command to which you want to assign the hotkey.

In the Commands list, select the command to which you want to assign the hotkey.

Type the hotkey or key combination that you want in the Press New Shortcut Key box and click [Assign]. If you press a key or key combination that is invalid, no key is displayed. You cannot assign key ESC, and combinations such as CTRL+ALT+DEL that are already being used by your operating system.

In the Current Keys list, select the hotkey you want to delete and click [Remove] to delete the key.

2. Reset Hotkey to the Default Value

To reset the hotkey to the default value, just click [Reset].

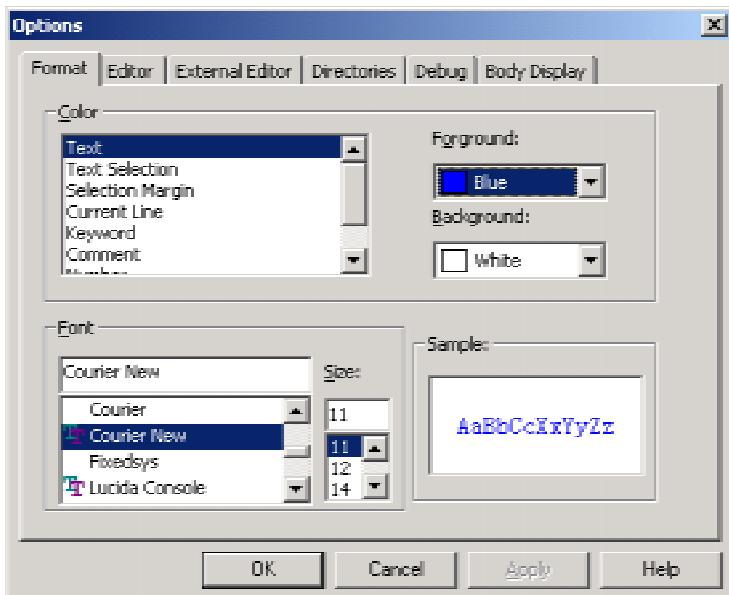
22.10.4 Editor

Options dialog box contains options which determine the shape and function of µ'nSP® IDE editor, the directories of included files and library-files used in compile and link mode, the foreground colors, text colors of different windows, and many others.

In Options dialog box, Format label can only be used in text editor. Many settings can influence the functions of editor and debugger directly. Current display mode and the status of debugger determinate the settings of other options. Developer Studio has four types of screen layouts:

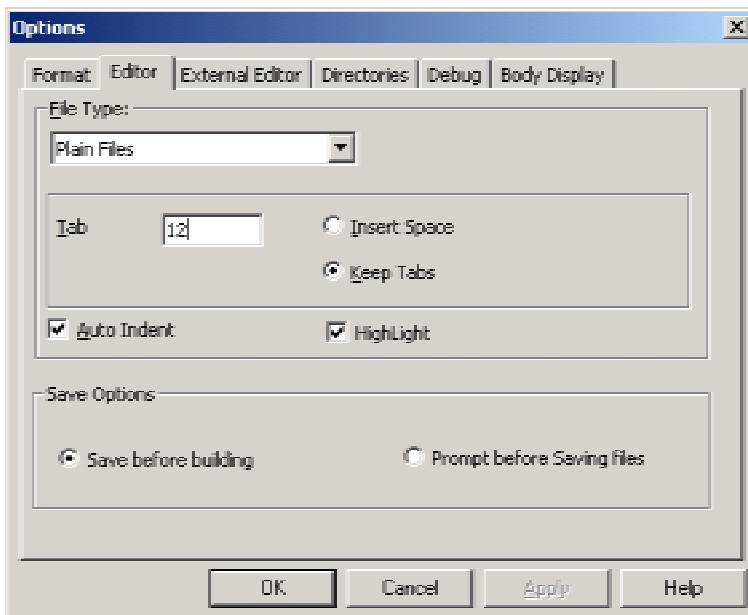
1. Standard layout in edit mode.
2. Full screen layout in edit mode.
3. Standard layout in debug mode.
4. Full screen layout in debug mode.

22.10.5 Format



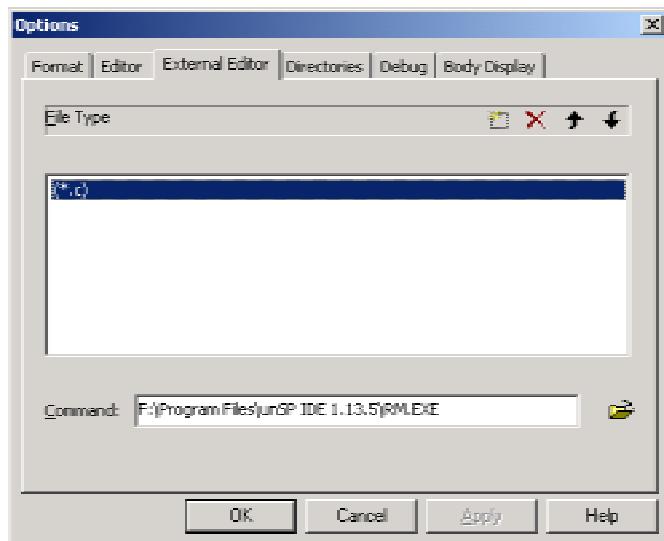
Different components (such as keyword, comment and number) can be set with different fonts, sizes and foreground, background color.

22.10.6 Editor



You can change the tab size, tab mode and indent mode with correspond file type. Save before building can be set to specify IDE save file before you build the project. Prompt before Saving files can be set to display prompt so that you can confirm that you want to save file.

22.10.7 External Editor

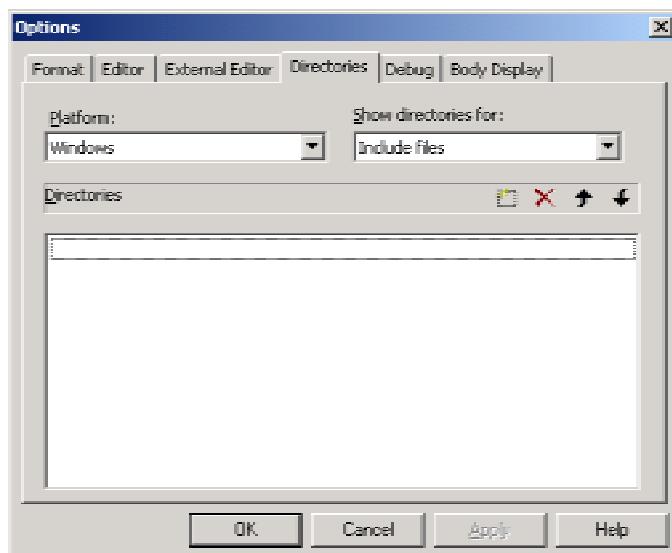


Set up the external editor in this window, and you can edit the selected file and active file with external editor by pressing Ctrl + E hot key.

Click  to create one item. Key in the correct file type, e.g., c file type(*.c) in the list box.

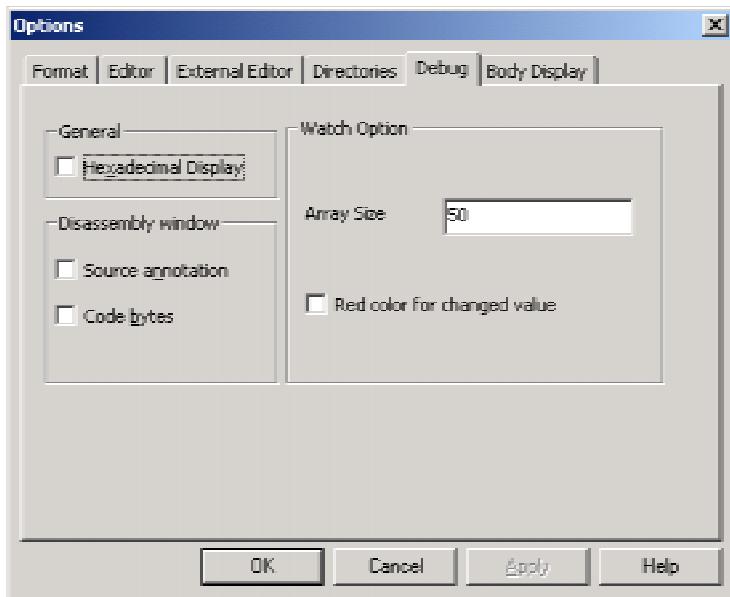
In Command area, you can select the editor for the file type you selected.

22.10.8 Directories



Select operation platform on which the directory path locate in Platform. Select included files and library files in Show directories to display the type of directory. In Directories list box list, you can find project files in hierarchical order, add directory, change the order to load all the necessary support files for your project.

22.10.9 Debug



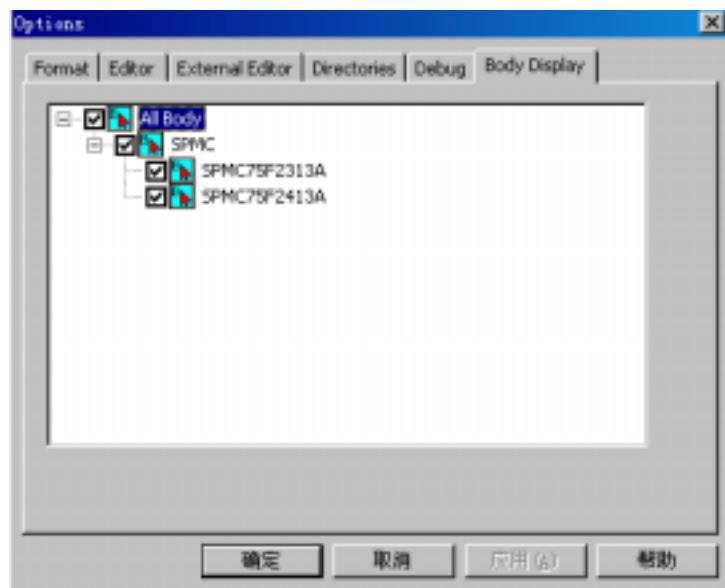
You can check Hexadecimal Display to display value in hexadecimal format and parse user input in hexadecimal in debug window and dialog.

Select Source annotation, the source code can be shown. Code bytes can be selected to show op code corresponding to each instruction.

Array Size is used to specify the array length in Watch window. When Color red for changed value is enabled, you can find the changed value is displayed in red in Watch window, but this option can affect the debug speed.

Red color for changed value is used to make the font color in Watch window changed into red when the value being changed.

22.10.10 Body Display



Specify supported body for the project.

22.11 Other Tool

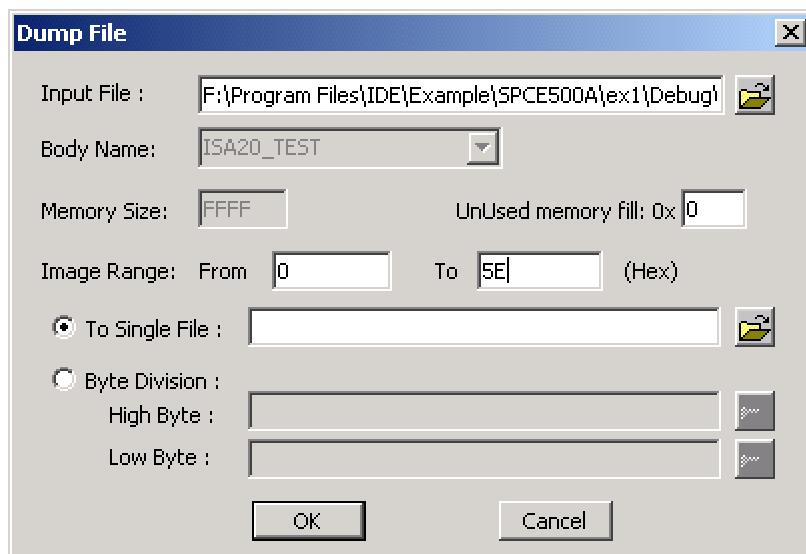
22.11.1 Dump File

The IDE can dump the content of the specified address range to one file/separated files with low/high byte division for ROM. To view the dump window, click [Tools]→[Dump File]. You can see one screen as below. Input the source file location, memory range, select the output file type, and input the output file name. Click [OK] to dump content to file (files).

Note: The input source file format only support S37 & TSK format.

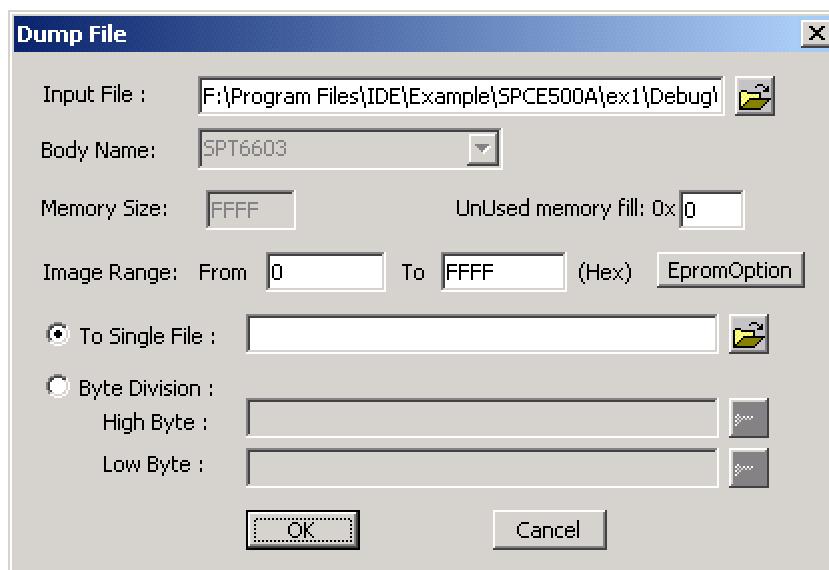
Different window should be shown according to the body type you selected.

For general body, e.g. select SPCE500A in Select Body dialog box.

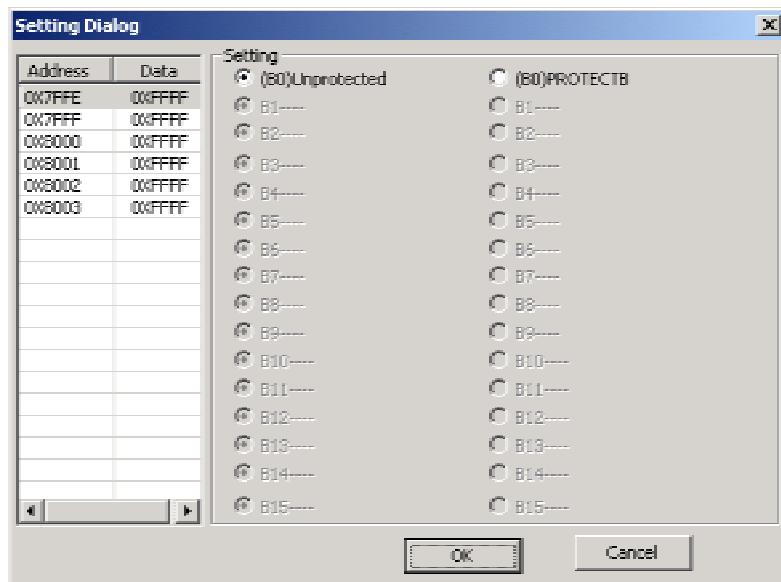


1. Input File: Specify the original binary file. It can be .tsk/.s37. When you select the .S37, UnUsed memory fill box can be displayed.
2. Body Name: Select body.
3. Memory Size: Show memory size.
4. UnUsed memory fill: Specify the value for unused memory.
5. Image Range: Specify the extraction range from input file.
6. To Single File: Extract to single file and you can specify the output file in the following edit window/button.
7. Byte Division: Extract to two files. One is low-byte data and the other is high-byte data. You can specify the output file in edit window.

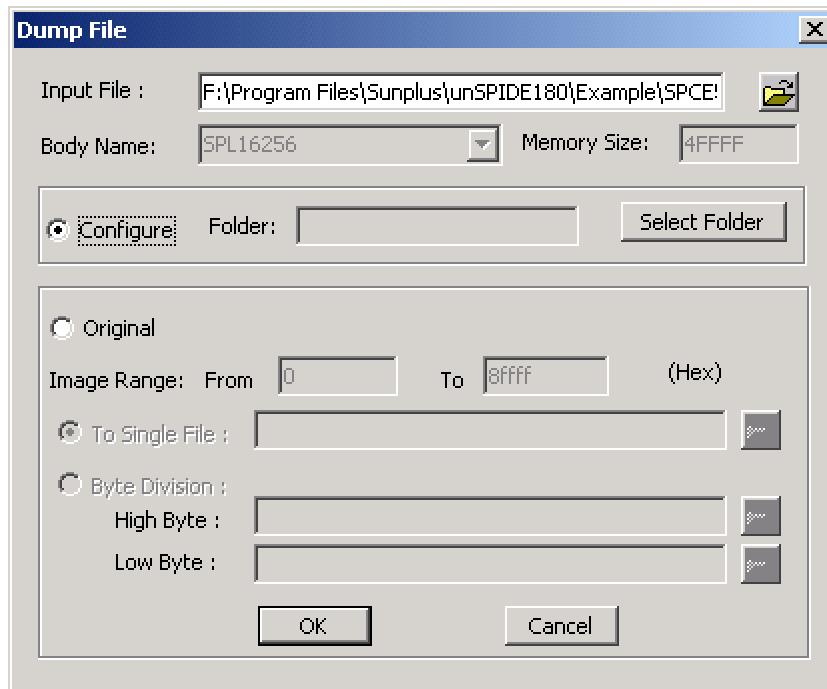
For the body with EPROM, e.g. select SPT6603 in Select Body dialog box.



Click [EpromOption], you can set bit value for specified address in the following window. Select the bit, you can write '1' to current bit. Unselect can be used to write '0' to the bit.



For the body with Chip Select, e.g. select SPL16256 in Select Body dialog box.

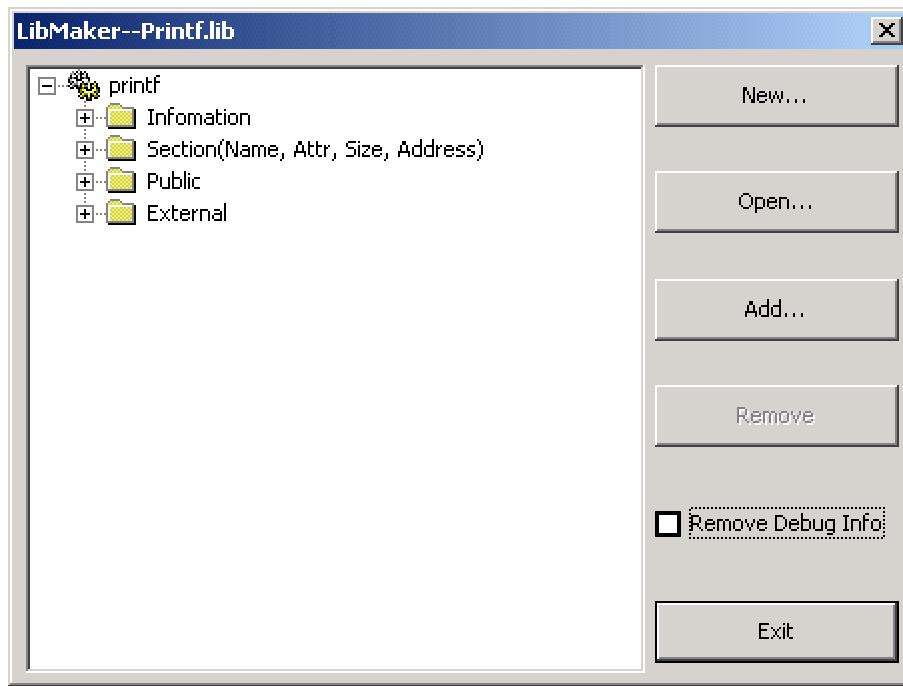


Configure: Select Configure to set folder for output file.

Original: Select Original to set extraction type and range.

22.11.2 LibMaker

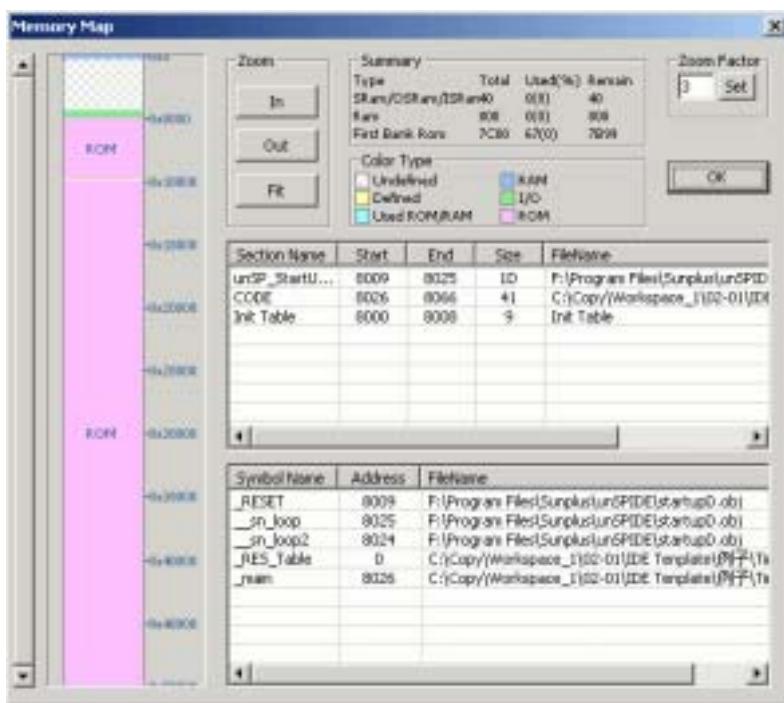
Click [LibMaker] in Tools menu, a Lib Maker Dialog will show as follow. You are allowed to create a new library/open an existed library to add/remove object file with it.



1. Click [New] to create a new library and click [Open] to open a exist library.
2. Click [Add] to add object files into library.
3. Select entries in lists and click [Remove] to remove selected object file.
4. Choose Remove Debug Info to make the debug info of the object file removed.
5. Click [Exit] to quit from this window and return to the main window.

22.11.3 MemoryMap

After clicking the [MemoryMap] in Tools menu, a memory-map dialog shows as follows. You can have an overview on the information of memory deployment, section, symbol, and statistics.



1. Click [In] to zoom in memory map.
2. Click [Out] to zoom out memory map.
3. Click [Fit] to zoom memory map to window size.
4. [Zoom Factory] set the zoom factor.
5. Click [OK] to quit from this window and return to the main window.

22.12 Hot Key

Click [Help]→[Keyboard] to see the hot key.

22.12.1 Toolbar Button

-  Create new document
-  Open document
-  Save active document
-  Saves all opening files
-  Cut the selection to the clipboard
-  Copy the selection to the clipboard
-  Insert from clipboard
-  Undo the last action
-  Redo the previously action
-  Show or hide the workspace window
-  Show or hide the output window
-  Manage currently opened windows
-  Find text in multiple files
-  Find the specified text in current file
-  Print the active document
-  Display program information,version number and copyright
-  Load program to device
-  Compile file
-  Build project
-  Stop build
-  Execute program
-  Download program
-  Debug program with simulator
-  Debug program with ICE
-  Select body
-  Arrange windows so they overlap
-  Arrange windows as non-overlapping tiles
-  Close all opened windows
-  Activate the next undocked window
-  Activate the previous undocked window
-  Break running program
-  Stop debugging the program
-  Restart the program

-  Step into the next statement
-  Step over the next statement
-  Step out the current statement
-  Run the program to current cursor
-  View history buffer
-  Set emulator
-  Insert or remove breakpoint
-  Remove all breakpoints
-  Show or hide the watch window
-  Show or hide the disassembly window
-  Show or hide the memory window
-  Show or hide the register window
-  Show or hide the command window
-  Show or hide the breakpoint window

23 SUNPLUS DMC Toolkit Introduction

Welcome to DMC (Digital Motor Control) Toolkit, a powerful tool authored from SUNPLUS. It features real time controlling and monitoring running state of two motors which maybe powered from inverter device. The user-friendly interface and pull-down menus assist you to make better use of the application easily and efficiently.

23.1 DMC Toolkit Features

23.1.1 Controlling System

Set motor control parameters for two independent motors, such as the parameters for speed rotation reference, speed working slope, speed control loop gain Ki, and Kp.

Provides up to sixteen parameters for user's application.

23.1.2 Monitoring System

Monitor control parameters in Monitoring window directly, and also you can monitor control parameters by observing wave in Charting window.

23.2 Installation

23.2.1 System Requirements

The tool of current version can be run on Windows98® or Windows2000®. The prepositional minimum requirements of system are here:

CPU Clock: 500M Hz

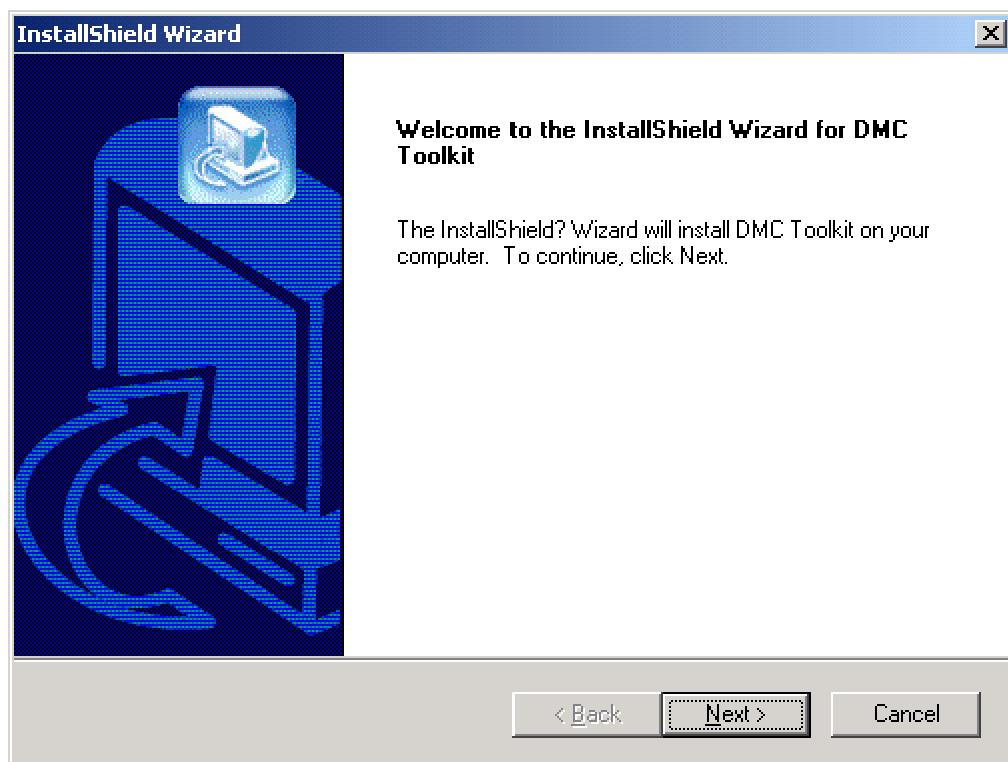
Capacity of Memory: 64 MB

Capacity of Hard Disk: 20 MB

UART Serial Port Interface

Take the following steps, and you will install this tool on your computer:

1. Execute the installation file.

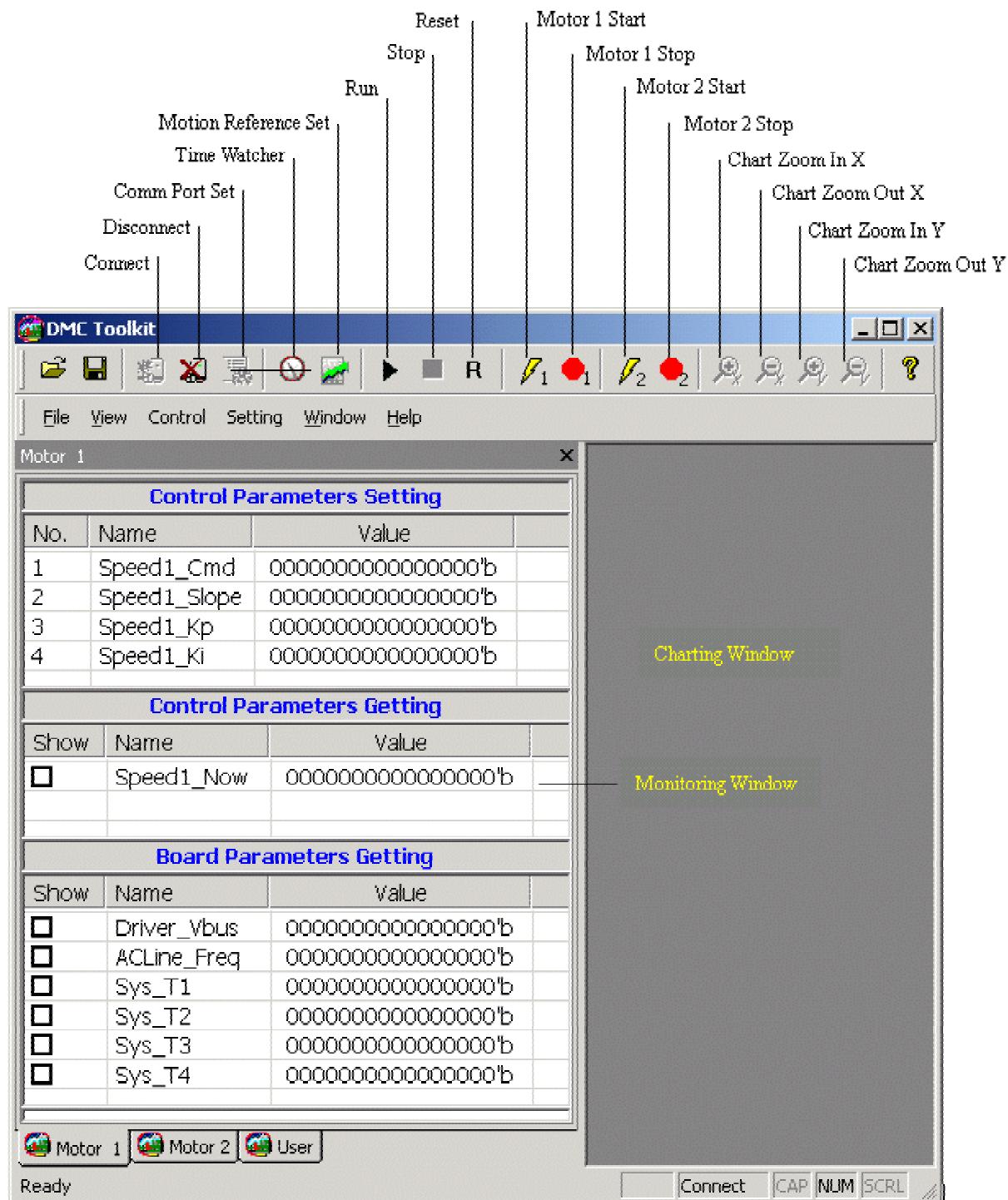


2. Follow the on-screen prompts, and the tool will be installed on your computer.

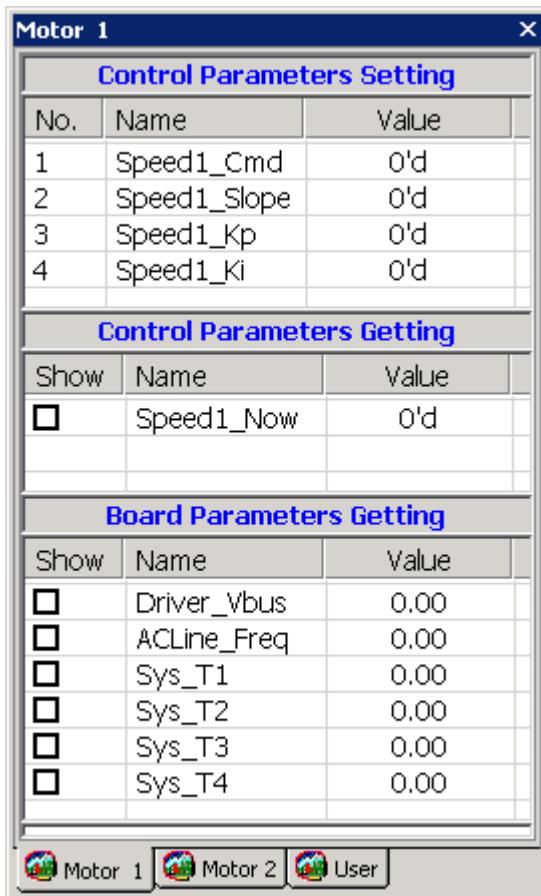
Note that your display setting of font size had better set at **Small Fonts** (Normal Size (96 dpi)) mode.

23.3 Looking Through DMC Toolkit Components

The user interface is the portion to display information and specify actions of your object in the application. By using the windows, menus, toolbars, you can control and monitor system. The tool consists of two parts: Monitoring window and Charting window.



23.3.1 Monitoring Window



Monitoring window is used to set and get parameter for system. It consists of three parts: **Motor1** window, **Motor2** window and **User** window. Each window includes three contents: **Control Parameters Setting**, **Control Parameters Getting**, and **Board Parameters Getting**. Meanings of parameters listed in **Monitoring** window will be described as follows.

Parameter Name	Attribute(R/W)	Description	Range
Speed1_Cmd	W	Set revolution for motor1 Bit15: direction bit	0 ~ ± 0xFFFF (unit: rpm)
Speed1_Slope	W	Set slope for motor1	0 ~ 0xFFFF (unit: rpm/sec)
Speed1_Kp	W	Set closed circuited adjusting parameter Kp for motor1	Greater than 0
Speed1_Ki	W	Set opened circuited adjusting parameter Ki for motor1	Greater than 0
Speed2_Cmd	W	Set revolution for motor2 Bit15: direction bit	0 ~ ± 0xFFFF (unit: rpm)
Speed2_Slope	W	Set slope for motor2	0 ~ 0xFFFF (unit: rpm/sec)

Parameter Name	Attribute(R/W)	Description	Range
Speed2_Kp	W	Set closed circuited adjusting parameter Kp for motor2	Greater than 0
Speed2_Ki	W	Set opened circuited adjusting parameter Ki for motor2	Greater than 0
Speed1_Now	R	Get running revolution of motor1 Bit15=1: negative revolution Bit15=0: position revolution	0 ~ ± 0x7FFF (unit: rpm)
Speed2_Now	R	Get running revolution of motor2 Bit15=1: negative revolution Bit15=0: position revolution	0 ~ ± 0x7FFF (unit: rpm)
Driver_Vbus	R	Get DC voltage	0 ~ 0xFFFF
ACLine_Freq	R	Get frequency	0 ~ 0xFFFF
Sys_T1	R	Get temperature of channel1 Bit15: sign bit	0 ~ ± 0x7FFF (unit:)
Sys_T2	R	Get temperature of channel2 Bit15: sign bit	0 ~ ± 0x7FFF (unit:)
Sys_T3	R	Get temperature of channel3 Bit15: sign bit	0 ~ ± 0x7FFF (unit:)
Sys_T4	R	Get temperature of channel4 Bit15: sign bit	0 ~ ± 0x7FFF (unit:)
User_W0	W	Set parameter area for programmer	
User_W1	W		
User_W2	W		
User_W3	W		
User_W4	W		
User_W5	W		
User_W6	W		
User_W7	W		
User_R0	R	Get parameter area for programmer	
User_R1	R		
User_R2	R		
User_R3	R		
User_R4	R		
User_R5	R		
User_R6	R		
User_R7	R		

You can set displaying format of parameter by selecting [View] → [Binary Format/Decadal Format/Hex Format] or right clicking in **Monitoring** window.

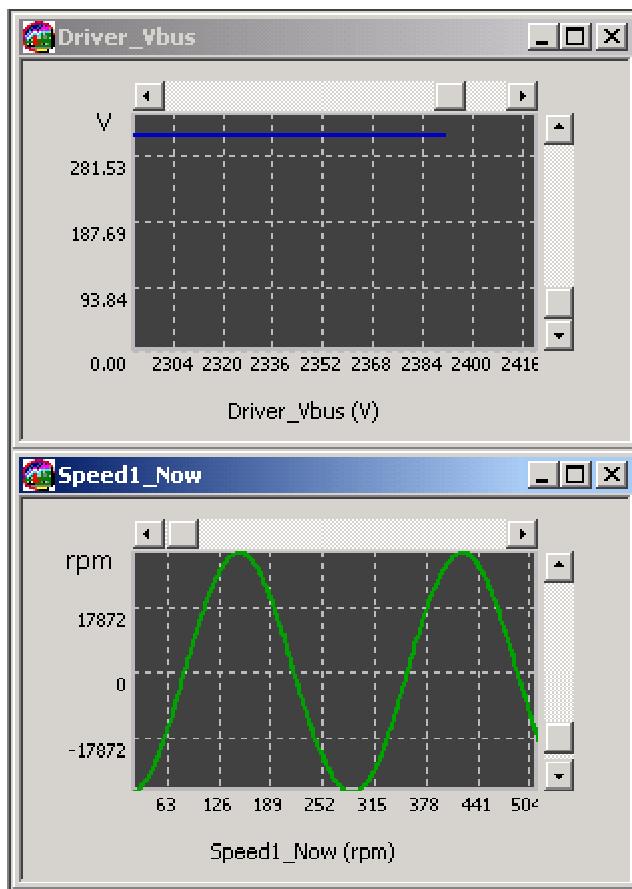
Set Q Value
 Binary Format
 Decadal Format
 Hex Format

Notes:

1. Option **Set Q Value** can only be used in **User** window.
2. Only parameter name in **User** window can rename by clicking parameter name and then pressing **Enter** or clicking other area.
3. The way that set parameter is the same as parameter name, and parameter can be modified after connecting system.

23.3.2 Charting Window

Charting window is used to display charted wave. You can option parameter in **Monitoring** window to add **Charting** window. Each **Charting** window that you can see is individual window, and you can also open or close it alone.

**Notes:**

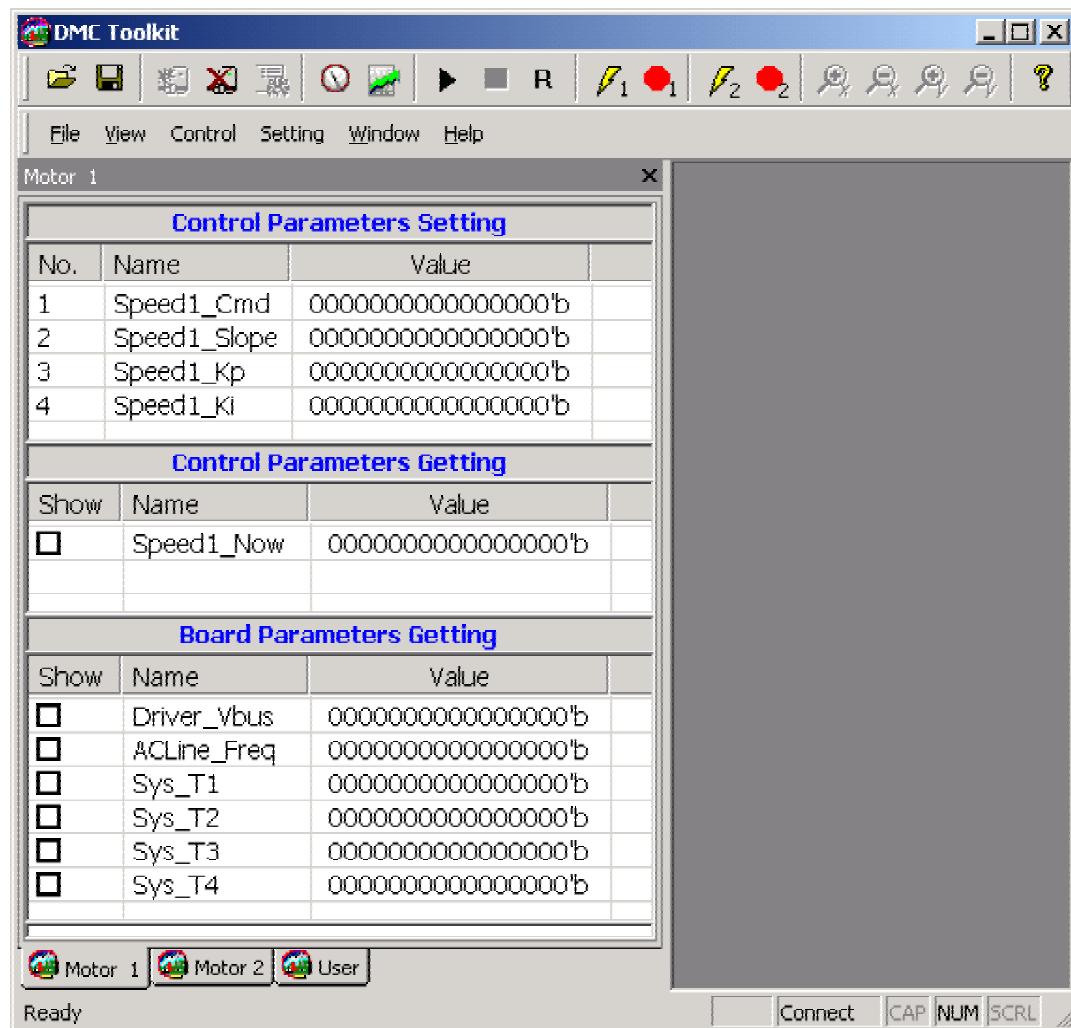
1. You can arrange each window by selecting arrange format that you want in **Window** menu.
2. You can change size of window by dragging mouse.

23.4 Quick Start

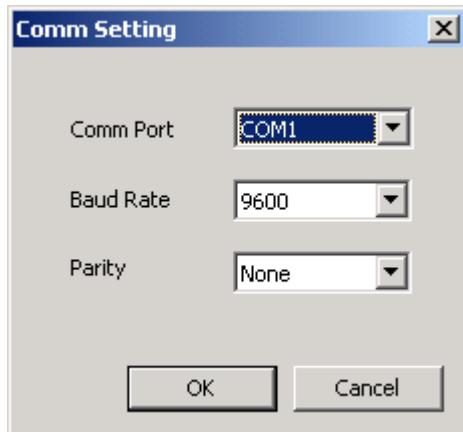
This section introduces an example of controlling and monitoring system, the example help you understand how to use DMC Toolkit.

23.4.1 Launching the DMC Toolkit

1. Select [Programs] → [SUNPLUS] → [DMC Toolkit] → [DMC Toolkit] to run the **DMC Toolkit**. A main window is displayed as follows:



2. Select [File] → [Comm Setting] or click  to open Comm Setting dialog box below.



3. Select [File] → [Connect] or click  to open serial interface.

Notes:

1. You can select [File] → [Disconnect] or click  to close serial interface.
2. If connecting error will appear a prompt dialog box as follows.



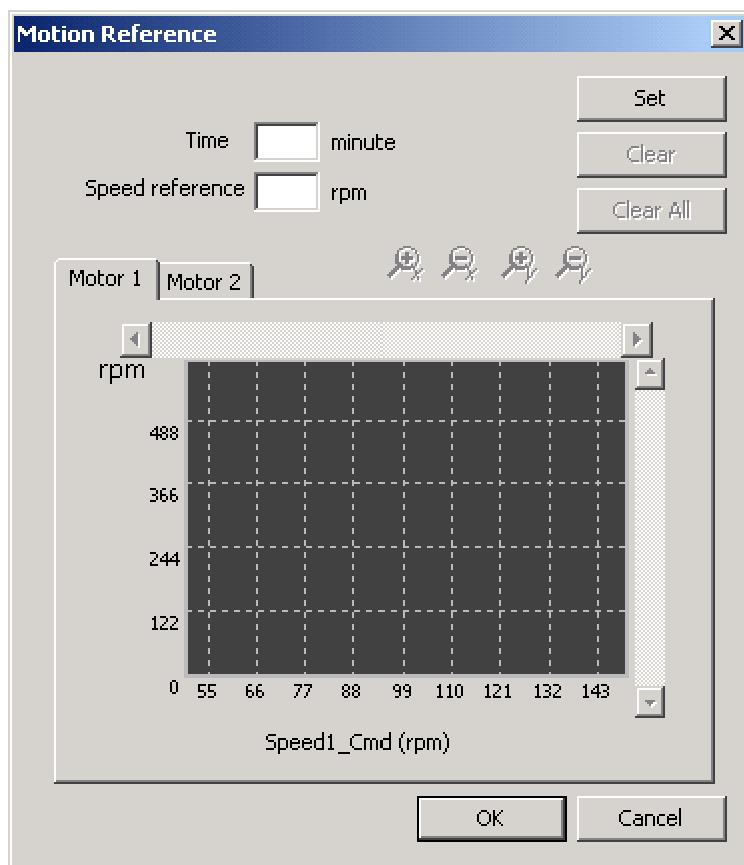
23.4.2 Controlling and Monitoring System

Select [Control] → [Start] or click  to start system.

Controlling System

Here two ways are provided for setting control parameters.

1. Set parameters in **Monitoring** window directly, refer to [Monitoring window](#).
2. Select [Setting] → [Motion Reference] or click  to open Motion Reference dialog box as follows.



Set: add one new point.

Clear: clear a set point.

Clear All: clear all points.

Note that **Motion Reference** dialog box is used to set parameter Speed_cmd.

Monitoring System

You can monitor control parameters in **Monitoring** window directly, and also you can monitor control parameters by observing wave in **Charting** window.

Notes:

1. You can select [Control] → [Stop/Reset] or click  to control system. After you press **Start**, control parameters and wave displayed in **Charting** window will be updated synchronously; when you press **Stop**, control parameters and wave displayed in **Charting** window stop update; when you press **Reset**, control parameters and wave displayed in **Charting** window will be reset.
2. You can export data file of wave in .txt format by right clicking on wave area in **Charting** window and then selecting [**Export...**].

23.4.3 Starting or Stopping Motor

Select [Control] → [Motor 1 Start/Motor 1 Stop/Motor 2 Start/Motor 2 Stop] or click     to start or stop motor.

Note: Starting motor is not synchronous with starting system, and the two actions have no priority.

23.4.4 Saving Project

Select [File] → [Save Project] or  to save the project.

You can also save the current project as a new project name or a different directory by clicking [File] → [Save As...].

23.4.5 Exiting DMC Toolkit

Select [File] → [Exit] to exit from DMC Toolkit.

23.5 Applying DMC Toolkit

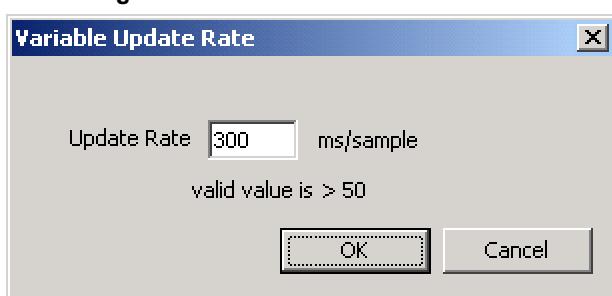
Here is the best place for you to get some hands-on experience about using this tool.

23.5.1 Opening an Existing File

Select [File] → [Open Project] or click  to open an existing file.

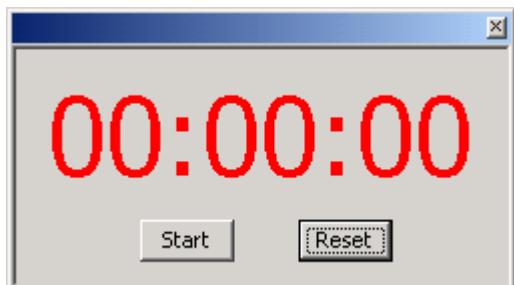
23.5.2 Setting for Variable Update Rate

Before starting system, you can select [Setting] → [Variable Update Rate...] to display a new dialog box to set update rate for wave displayed in Charting window and control parameters displayed in Monitoring window.



23.5.3 Starting Time Function

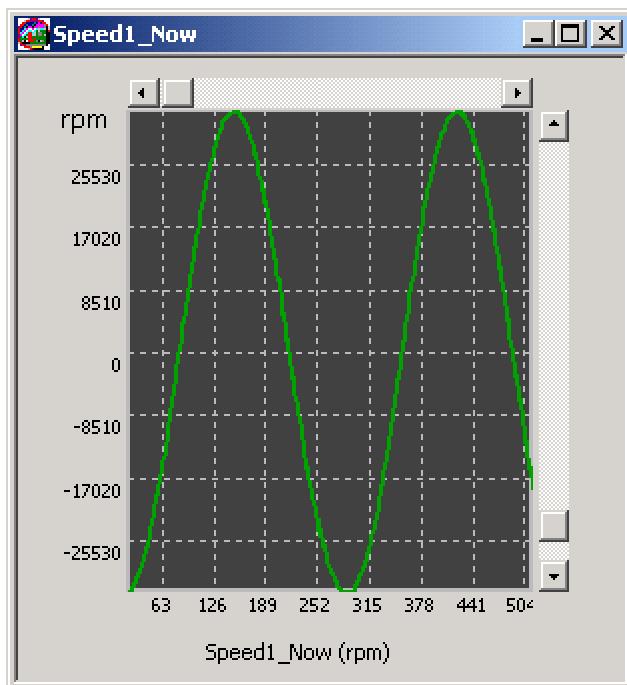
Select [View] → [Time Watcher] to open a dialog box as follows.



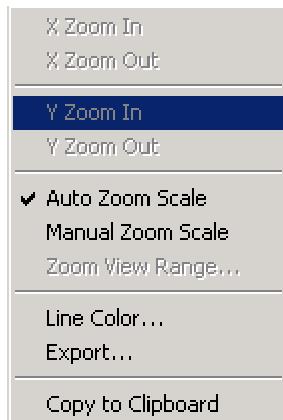
When you press **Start** button, time function will start and **Start** change into **Stop**; when you want to stop the function, you can press **Stop** button.

Note that the function has no relationship with Start/Stop of motors.

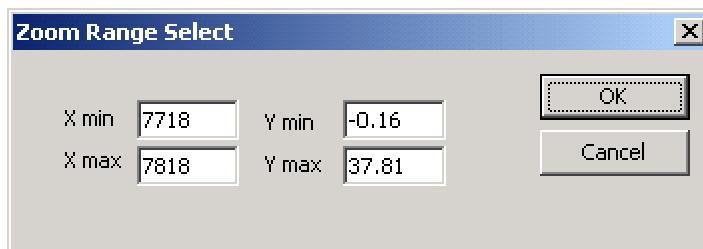
23.5.4 Editing Charting Window



Right click on wave area in **Charting** window will display hotkey menu as follows.



In Manual Zoom Scale mode, you can select [Zoom View Range...] to open [Zoom Range Select] dialog box to set view range that you want.



Line Color...: you can choose color for wave, and the initial setting is bottle green.

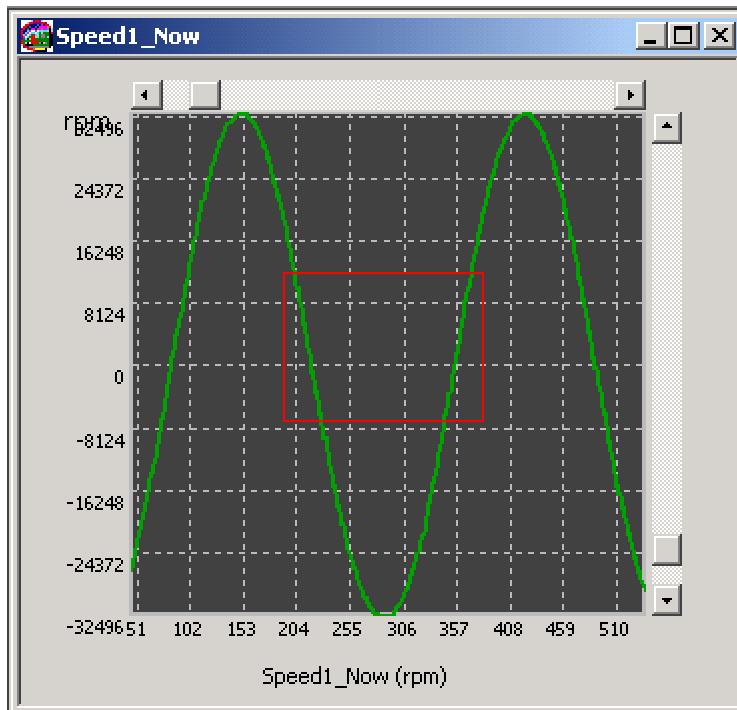


Export: export data file of wave in .txt format.

Copy to Clipboard: copy wave to clipboard.

Notes:

1. You can change chart background by selecting [Setting] → [Chart Background...].
2. You can zoom in/zoom out wave by using  displayed in toolbar.
3. You can zoom in wave by dragging and dropping mouse, and the selected wave (in the red area) will be displayed in current window. But the function is only used in Manual Zoom Scale mode.



23.6 Accelerator

Ctrl+O	Open Project
Ctrl+S	Save Project
Ctrl+T	Time Watcher
Ctrl+G	Start
Ctrl+P	Stop
Ctrl+R	Reset
Ctrl+M	Motion Reference

23.7 Technical Support

DMC Toolkit is designed to be easy to use, however, on occasion you may need some explanation or help with an issue. In most cases, you'll find the help you need right in your computer; this help file, and DMC Toolkit user manual should contain the answers to most of your questions. In addition, you'll find extensive support material available around-the-clock at our site on the World Wide Web. If you still experience difficulties, our technical support staff is available via Email. We recommend sending your technical questions electronically (Email) for the most efficient handling of your request.



For furthermore tech support, you can contact with your SUNPLUS representative.

Web Site Support: <http://www.sunplus.com.tw/>

Email Support: mike@sunplus.com.tw

24 APPENDIX A. Reserved Words

The followings are reserved words for µ'nSP™ except the assembly language commands are not listed here.

1. Words with prefix of “_sn_” are reserved.
2. _BREAK, _FIQ, _RESET, _IRQ0, _IRQ1, _IRQ2, _IRQ3, _IRQ4, _IRQ5, _IRQ6, _IRQ7, __subf2, __blockcopy, __cmpf2, __common, __cvf2i1, __cvf2i2, __cvf2u1, __cvf2u2, __cvi1f2, __cvi2f2, __divf2, __divi1, __divi2, __divu1, __divu2, __lshiu1, __lshiu2, __modi1, __modi2, __modu1, __modu2, __mulf2, __muli2, __mulu1, __mulu2, __negf2, __rshi1, __rshi2, __rshu1, __rshu2, __addf2