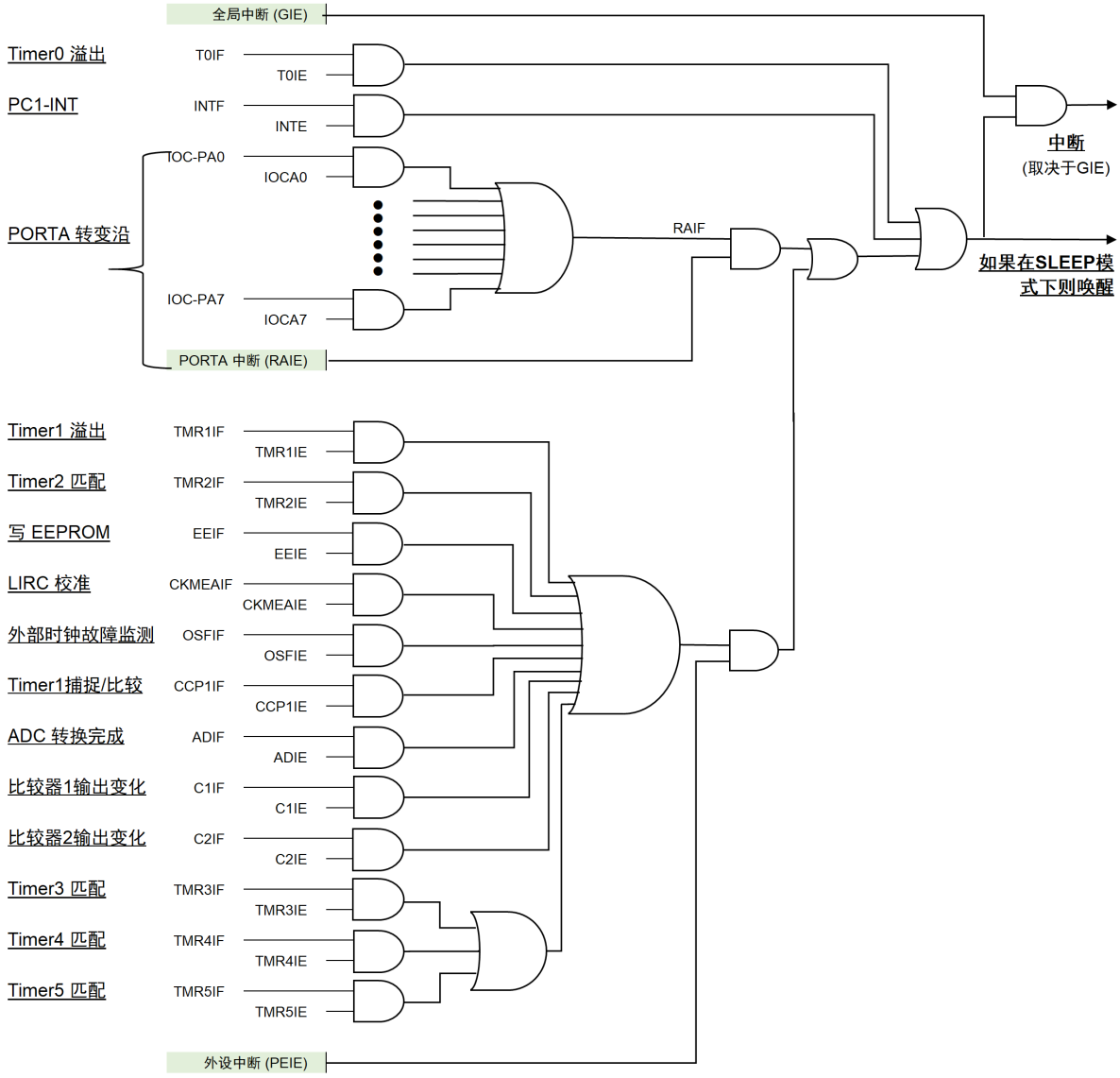
FT61F02X

INT Application note

Table of contents

FT61F02x INTapplication

1.interrupt(INTERRUPTS)



注：由于系统时钟在休眠期间暂停，只有不依赖于系统时钟的外设会将器件从休眠中唤醒

picture1-1 Block Diagram of Interrupt Structure

CPUsupport15interrupt sources, divided into2Group:

1)non-peripheral interrupt (Timer0andI/O)

- Timer0overflow
- PC1−INT (automatic rising or falling edge interrupt)
- PORTAPort change interrupt (software controlled)

2)peripheral interrupt

- Timer1overflow

- Timer2andPR2match
- DATA EEPROMwrite finished
- LIRCandHIRCCross Calibration Complete
- Fail-Safe Clock Monitor
- Timer1capture/compare
- ADCconversion complete
- Comparators1 /Comparators2output change
- Timer3 / Timer4 / Timer5match

with otherTimersdifferent,WDTOverflow does not generate an interrupt. ExceptI/OFor other interrupts, please refer to the corresponding chapters.

When an interrupt is generated,PCJump and execute "Interrupt Service Routine (ISR)". There are multiple layers of control for disabling/enabling interrupts:

- Each interrupt source has its own independent interrupt enable bit:T0IE, INTE, IOCAx, TMRxIE(x=1,2,3,4,5), EEIE, CKMEAIE, CxIE(x=1,2), OSFIE, ADIE, CCP1IE.
- 8indivualPAxThe interrupt inputs share a port interrupt enable bit:PAIE (PORTA Interrupt Enable).
- Peripheral interrupts have a global interrupt enable bit:PEIE (PEripheral Interrupt Enable).
- If all the above control bits are turned off, wake-up from sleep will not be performed.
- All interrupts are controlled by the global interrupt enable bits:GIE (Global Interrupt Enable). Unlike other enable bits,When the global interrupt enable bit is turned off, wake-up from sleep is still allowed .
- Turning off the interrupt enable bit does not affect the setting of the interrupt flag bit .
- Timer0Interrupts cannot wake up from sleepCPU.

The interrupt processing sequence is as follows:

- auto configuration"GIE = 0", thereby turning off the interrupt.
- The return address is pushed onto the stack, the program pointerPCload0x0004address.
- after an interruption1-2instruction cycle, jump to "interrupt service routine (ISR)"Start handling interrupts.
- Execute "return from interrupt(RETI)"command exitISR.existRETIAll interrupt flag bits must be cleared before .
- whenISRwhen finished,PCreturn to the address before the interrupt, if theSLEEPmode, returns toSLEEPThe address immediately following the instruction.
- in executionRETIAutomatically set when "GIE = 1", thereby enabling the interrupt.

Note: During the interrupt process, only returnPCAddresses are automatically saved on the stack. If the user needs to save other important register values (such as W, STATUSregisters, etc.), these values must be correctly written into the temporary registers through the instruction, it is recommended to useGPRthe last of16indivualbytesas temporary registers because allbankshare this16indivualbytes, without switchingbankto save code.

1.1. Summary of Interrupt Related Registers

name	address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	reset value (RW)
INTCON	0x0B	GIE	PEIE	T0IE	INTE	PAIE	T0IF	INTF	PAIF	0000 0000
PIE1	0x8C	EEIE	CKMEAIE	–	C2IE	C1IE	OSFIE	TMR2IE	TMR1IE	0000 0000
PIR1	0x0C	EEIF	CKMEAIF	–	C2IF	C1IF	OSFIF	TMR2IF	TMR1IF	0000 0000
PIE2	0x8D	–	–	–	–	–	–	ADIE	CCP1IE	------00
PIR2	0x0D	–	–	–	–	–	–	ADIF	CCP1IF	------00
PWM3CR0	0x10F	P3INTS	P3PER[2:0]			P3CKSRC[2:0]			wxya	0000 0000
PWM3CR1	0x110	P3EN	P3POL	TMR3PS[2:0]			TMR3ON	TMR3IE	TMR3IF	0000 0000
PWM4CR0	0x115	P4INTS	P4PER[2:0]			P4CKSRC[2:0]			wxya	0000 0000
PWM4CR1	0x116	P4EN	P4POL	TMR4PS[2:0]			TMR4ON	TMR4IE	TMR4IF	0000 0000
PWM5CR0	0x11B	P5INTS	P5PER[2:0]			P5CKSRC[2:0]			QUR	0000 0000
PWM5CR1	0x11C	P5EN	P5POL	TMR5PS[2:0]			TMR5ON	TMR5IE	TMR5IF	0000 0000
OPTIONS	0x81	/PAPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
TRISA	0x85	PORTAdirection control								1111 1111
IOCAs	0x96	PORTAPort Change Interrupt Settings								0000 0000

surface1-1 Interrupt-related register addresses and default values

name	state		register	address	reset value
GIE	global interrupt	1 =Enable (PEIE,Independent enable bits for each interrupt apply) 0 =global shutdown (wake up is not affected)	INTCON[7]	0x0B 0x8B 0x10B	RW−0
PEIE	Total Peripheral Interrupt	1 =Enable (each interrupt independent enable bit applies) 0 =closure (no wakeup)	INTCON[6]		RW−0
T0IE	Timer0overflow interrupt	1 =Enable 0 =closure (no wakeup)	INTCON[5]		RW−0
INTE	PC1−INTExternal Interrupt		INTCON[4]		RW−0
PAIE	PORTAport change interrupt		INTCON[3]		RW−0
T0IF	Timer0overflow interrupt flag	1 = Yes (latch) 0 =no	INTCON[2]		RW−0
INTF	PC1−INTexternal interrupt flag		INTCON[1]		RW−0
PAIF	PORTAport change interrupt flag		INTCON[0]		RW−0

surface1-2INTCONregister

name	state		register	address	reset value
EEIE	EEwrite complete interrupt	1 =Enable 0 =closure (no wakeup)	PIE1[7]	0x8C	RW−0
CKMEAIE	LIRCandHIRCCross Calibration Done Interrupt		PIE1[6]		RW−0
C2IE	Comparators2to interrupt		PIE1[4]		RW−0
C1IE	Comparators1to interrupt		PIE1[3]		RW−0
OSFIE	External Oscillator Fail Interrupt		PIE1[2]		RW−0
TMR2IE	Timer2andPR2match break		PIE1[1]		RW−0
TMR1IE	Timer1overflow interrupt		PIE1[0]		RW−0
ADIE	ADCconversion complete interrupt		PIE2[1]	0x8D	RW−0
CCP1IE	CCP1capture/match interrupt		PIE2[0]		RW−0

surface1-3PIExregister

name	state		register	address	reset value
EEIF	EEPROMwrite complete flag	1 = Yes (latch) 0 =no	PIR1[7]	0x0C	RW-0
CKMEAIF	LIRCandHIRCCross-calibration complete flag		PIR1[6]		RW-0
C2IF	Comparators2interrupt flag		PIR1[4]		RW-0
C1IF	Comparators1interrupt flag		PIR1[3]		RW-0
OSFIF	External Oscillator Fault Flag Bit		PIR1[2]		RW-0
TMR2IF	Timer2andPR2match flag		PIR1[1]		RW-0
TMR1IF	Timer1overflow flag		PIR1[0]		RW-0
ADIF	ADCconversion complete flag		PIR2[1]	0x0D	RW-0
CCP1IF	CCP1A capture/match occurred flag		PIR2[0]		RW-0

surface1-4 PIRxregister

name	state		register	address	reset value
P3INTS	Timer3interrupt select bit	1 = TMRxandPRxmatch break 0 = TMRxoverflow interrupt (x=3, 4, 5)	PWM3CR0[7]	0x10F	RW-0
P4INTS	Timer4interrupt select bit		PWM4CR0[7]	0x115	RW-0
P5INTS	Timer5interrupt select bit		PWM5CR0[7]	0x11B	RW-0
TMR3IE	Timer3interrupt enable bit	1 =Enable 0 =closure (no wakeup)	PWM3CR1[1]	0x110	RW-0
TMR4IE	Timer4interrupt enable bit		PWM4CR1[1]	0x116	RW-0
TMR5IE	Timer5interrupt enable bit		PWM5CR1[1]	0x11C	RW-0
TMR3IF	Timer3interrupt flag	1 =match/overflow (latch) 0 =no match/no overflow	PWM3CR1[0]	0x110	RW-0
TMR4IF	Timer4interrupt flag		PWM4CR1[0]	0x116	RW-0
TMR5IF	Timer5interrupt flag		PWM5CR1[0]	0x11C	RW-0

surface1-5 Timer3/4/5interrupt register

name	state	register	address	reset value
/PAPU	PORTApull up 1 =global shutdown 0 =Depend onWPUAcontrol	OPTION[7]	0x81	RW-1
INTEDG	PC1interrupt edge 1 =rising edge 0 =falling edge	OPTION[6]		RW-1
TRISA	PORTA I/ODigital output (direction control) 1 =input (close digital output) 0 =Close pull-up/pull-down	TRISA[7:0]	0x85	RW-11111111
IOCAs	PORTAport change interrupt 1 =Enable 0 =closure	IOCA[7:0]	0x96	RW-00000000

surface1-6OPTIONS,TRISAandIOCAsregister

1.2. PC1-INTandPORTAport change interrupt

name	PC1-INT	PORTAport change interrupt
number of channels	onlyPC1	PA0 – PA7 (up to8channels)
I/Oset up	TRISC[1] = 1; ANSEL[4] = 0; CMCON0[2:0] = 000	TRISA[x] = 1; ANSEL[x] = 0; CMCON0[2:0] = 000
other settings	INTEDG, INTE, GIE, INTF	IOCA, PAIE, GIE, PAIF
trigger	Rising edge or falling edge (choose one of the two)	0 → 1or1 → 0
Need software monitoring?	no	need

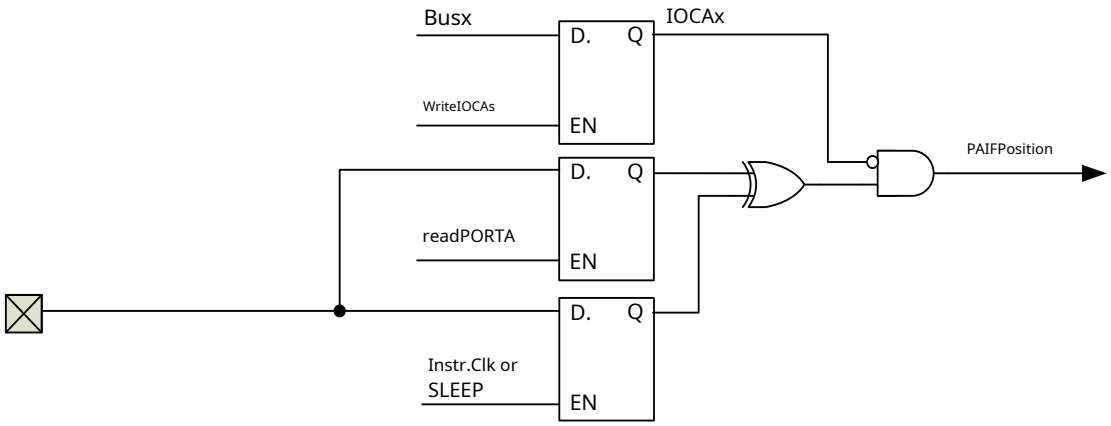
surface1-7PC1-INTandPORTAThe difference between port change interrupts

PC1-INTandPORTAPort change interrupts are both externalI/Ointerruption. If set correctly,PC1-INTwill run in the background without supervision.PORTAPort change interrupts require continuous software monitoring. forPORTAPort change interrupt:

1.Latch the input register value into the port change interrupt latch (by readingPORTA).

2.When the input level changes, the difference between the input register value and the latch value sets thePAIF.

3.The latching process of the input register (ie readPORTAprocess) will update the reference level used for comparison, if thePAIFRead immediately after asserting PORTAThe trigger condition of the port change interrupt can be cleared. When the port mismatch event no longer exists,PAIFCan be cleared by command.



picture1-2PORTAtransition edge interrupt

2.Application example

```
//****************************************************
********** /*file name:TEST_61F02x_INT.c
* Features:     FT61F02x-INTDemo
*IC:            FT61F023 SOP16
* Crystal:      16M/2T
* illustrate:   in the programDemoPortOut(PA3)output100frame50HzThe duty cycle is50%After the square wave, MCUGo
*               to sleep, waiting for the occurrence of external interrupt;
*               When the external interrupt is triggered, repeat the above process;
*
*                 FT61F023 SOP16
*                 ------------------------------
* VDD----------|1(VDD)       (VSS)16|----------GND
* NC-----------|2(PA7)       (PA0)15|----------NC
* NC-----------|3(PA6)       (PA1)14|-- ----------NC
* NC-----------|4(PA5)       (PA2)13|----------NC
* NC-----------|5(PC3)       (PA3)12|--DemoPortOut
* NC-----------|6(PC2)       (PC0)11|----------NC
* NC-----------|7(PA4)       (PC1)10|----------INT<--to interrupt
* NC-----------|8(PC5)       IN (PC4)09|----------NC
*                 ------------------------------
*/
//************************************************** ***********
# include "SYSCFG.h"
//************************Macro definition ***************************
#define   unchar            unsigned char
#define   Demo Port Out     PA3

unchar    FCount;
/*------------------------------------------------ -
* Function name:interrupt ISR
* Features:   PC1Interrupt handler
* enter:      none
* output:     none
  ------------------------------------------------------
----- * / void interrupt ISR(void)
{
    if(INTE && INTF)
    {
        INTF = 0;                   //clearPC1 INTflag bit //Temporarily
        INTE = 0;                   bannedINTto interrupt
    }
}
```

```
/*---------------------------------------------- ----
* Function name:POWER_INITIAL
* Features:  Power-on system initialization
* enter:  none
* output:  none
  ------------------------------------------------
--------- * / void POWER_INITIAL (void)
{
    OSCCON = 0B01110001;          //IRCF=111=16MHz/2=8MHz,0.125μs//Temporarily
    INTCON = 0;                   disable all interrupts
    PORTA = 0B00000000;
    TRISA = 0B00000000;           //PAinput Output0-output1-enter
    PORTC = 0B00000000;
    TRISC = 0B00000010;           //PCinput Output0-output1-enter
                                  //PC1-enter
    WPUA = 0B00000000;            //PAPort pull-up control1-pull up0-close pull //
    WPUC = 0B00000000;            PCPort pull-up control1-pull up0-close pull

    OPTION = 0B00001000;          //Bit3=1, WDT MODE, PS=000=WDT RATE 1:1
    MSCKCON = 0B00000000;
    //Bit6->0,prohibitPA4,PC5Regulated output
    //Bit5->0,TIMER2the clock isFosc //Bit4->0,
    prohibitLVR
    CMCON0 = 0B00000111;          //turn off the comparator,Cxfor numbersIOmouth
}
/*---------------------------------------------- ----
* Function name:Delay Us
* Features:  Short delay function --16M-2T--probably fast1%about.
* enter:  TimeDelay time length Delay time lengthTime μs none
* output:
  ------------------------------------------------
--------- * / void DelayUs(unsigned char Time)
{
    unsigned char a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}
/*---------------------------------------------- ----
* Function name:DelayMs
* Features:  short delay function
* enter:  TimeDelay time length Delay time lengthTime ms none
* output:
```

```
  ------------------------------------------------
--------- * / void DelayMs(unsigned char Time)
{
    unsigned char a, b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197);                //quick1%
        }
    }
}
/*------------------------------------------------- -
* Function name:INT_INITIAL
* Features:   interrupt initialization function
* enter:    none
* output:   none
  ------------------------------------------------
----- */ void INT_INITIAL(void)
{
    TRISC1 = 1;                          //PC1-enter
    ANSEL = 0B00000000;                  //Disable interrupt pin analog input function

    INTEDG = 1;                          //OPTION,INTEDG=1; PC1 INTtrigger on rising edge //
    INTF = 0;                            clearPC1 INTinterrupt flag //EnablePC1 INTto interrupt
    INTE = 1;
}
/*------------------------------------------------- -
* Function name:main
* Features:   main function
* enter:    none
* output:   none
  ------------------------------------------------------
*/ void main()
{
    POWER_INITIAL();                     //system initialization

    while(1)
    {
        for(FCount=0;FCount<100;FCount++)//output100subwaveform {

            DemoPortOut = 1;
            DelayMs(10);                 //10ms
            DemoPortOut = 0;
```

```
            DelayMs(10);
        }
        INT_INITIAL();                  //Initialize external interrupt
        GIE =   1;                      //open total interrupt
        SLEEP();                        //to sleep
    }
}
```

Contact information

Fremont Micro Devices Corporation

5−8, 10/F, Changhong Building Ke-Ji
Nan 12 Road, Nanshan District,
Shenzhen, Guangdong, PRC 518057

Tel: (+86 755) 8611 7811
Fax: (+86 755) 8611 7810

Fremont Micro Devices (HK) Limited

16, 16/F, Block B, Veristong Industrial Centre, 34−36 Au Pui
Wan Street, Fotan, Shatin, Hong Kong SAR

Tel: (+852) 2781 1186
Fax: (+852) 2781 1144

http://www.fremontmicro.com