Translated from Chinese (Simplified) to English - www.onlinedoctranslator.com

FT61F02X

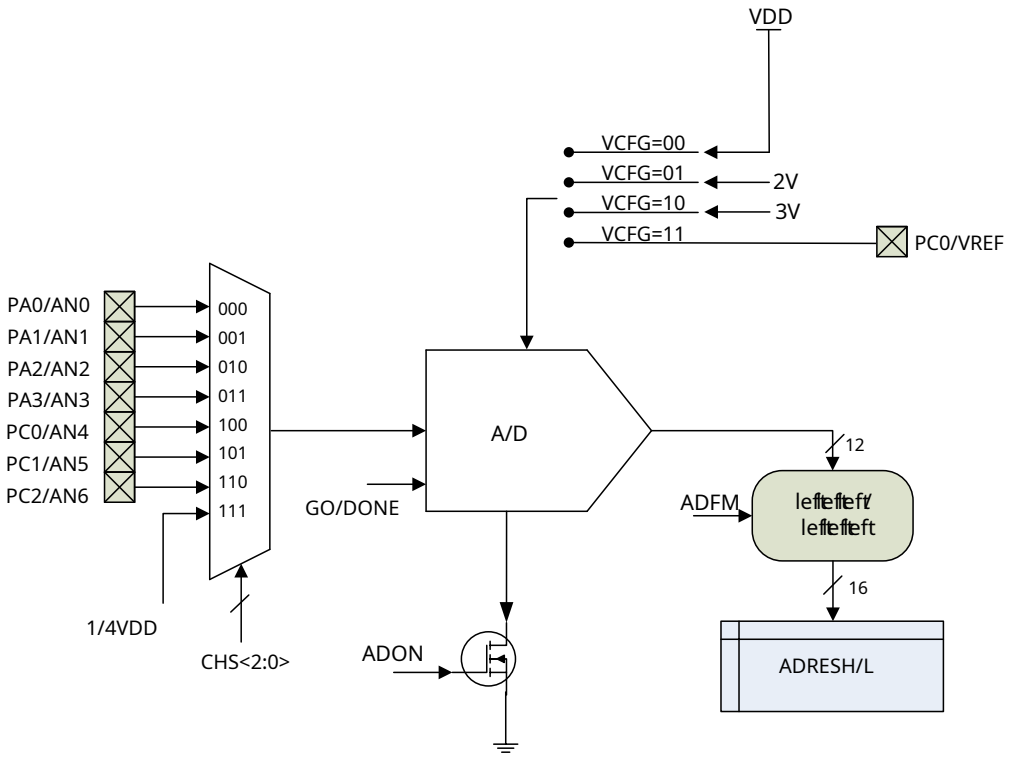
ADC Application note

Table of contents

FT61F02x ADCsapplication

1. 10bitA/D Converter (ANALOG TO DIGITAL CONVERTER, ADC)

ADCmodule converts an analog input signal into10-bitdigital signal.ADCCan run at different clock speeds and at up to 500 kHzclock speed (i.e.43 kHzthe sampling rate,23 μs/sampling) still has a true10-bitprecision.



picture1-1 ADCStructure diagram

The analog input signal can be7indivualI/O (ANx)one of the channels or1internal channels (Internal 1/4VDD).ADCby instruction orECCP Special event trigger fires. in the trigger andADCA delay can be added between samples.

whenADCWhen the conversion is complete, the corresponding interrupt flag bit is set and can trigger an interrupt and/or wake-up from sleep.

ADCReference voltage ($V_{ADC-REF}$)Selected by command asV_{DD},2an internal reference voltage (2V, 3V)one of, or throughI/OInput external reference voltage.

ADCCalibration is not required. in addition,ADCThe conversion process runs in the background during`CPU`Other commands can be executed.

ifADCneed to be inSLEEPkeeps running, and its conversion clock source isSysclkor its frequency division, you need to additionally enable the selected clock source asSysclkofTimers, to enable the system clockSysclkexistSLEEPkeep running. whenADCThe clock source forLIRCwhen, enter SLEEPbackLIRCwill be turned on automatically.

whenADCConfigured as a hardware trigger (ECCPspecial event trigger),GO/DONEAsserted and started directly by a hardware trigger eventA/Dchange change, set by softwareGO/DONEwill be ignored.ECCPSpecial event triggers can occur periodically without software interventionADC Measurement. When a trigger event occurs,Timer1The counter is reset to zero. The use of special event triggers does not ensure properADCtiming, the user must ensure that theADCtiming requirements. For more information see**No.**Error! Reference source not found.**Festival**"enhanced capture/compare/PWM (ECCP)".

1.1. ADCsSummary of related registers

name	state		register	address	reset value
GIE	global interrupt	1 =Enable (PEIE, ADCIEBe applicable) 0 =global shutdown (wake up is not affected)	INTCON[7]	0x0B 0x8B 0x10B	RW−0
PEIE	Total Peripheral Interrupt	1 =Enable (ADCIEBe applicable) 0 =closure (no wakeup)	INTCON[6]		RW−0
ADIE	ADCconversion complete interrupt	1 =Enable 0 =closure (no wakeup)	PIE2[1]	0x8D	RW−0
ADIF	ADCConversion Complete Interrupt Flag bit	1 = Yes (latch) 0 =no	PIR2[1]	0x0D	RW−0

surface1-1 ADCInterrupt Enable and Status Bits

name	state	register	address	reset value
ADRESL	ADCThe least significant bits of the conversion result (LSB) ADFM=0: ADRESL[7:6] =Low2bits (the rest are "0") ADFM=1; ADRESL[7:0] =Low8bit	ADRESL[7:0]	0x9E	RW−xxxx xxxx
ADRESH	ADCconversion result high significant bit (MSB) ADFM=0: ADRESH[7:0] =high8bit ADFM=1: ADRESH[1:0] =high2bits (the rest are "0")	ADRESH[7:0]	0x1E	RW−xxxx xxxx
ADFM	A/DConversion result format (see "ADRESH") 1 =right align 0 =align left	ADCON0[7]	0x1F	RW−0
VCFG	$V_{ADC-REF}$(reference voltage) 00 =V_{DD} 01 =internal 2V 10 =internal3V 11 =External reference voltage (I/O) Note:PC0Input as external referenceV_{REF}Must be set as an analog pin	ADCON0[6:5]		RW−00
CHS	ADCAnalog input channel 000 =AN0 001 = AN1 010 = AN2 011 = AN3 100 = AN4 101 = AN5 110 = AN6 111 = (internal1/4V_{DD})	ADCON0[4:2]		RW−000
GO/DONE	ADCConversion Enable and Status Bits 1 =by software,ECCP1start upA/Dconvert (automatically cleared after conversion) 0 =Conversion done/No conversion in progress	ADCON0[1]		RW−0
ADON	1 = ADCEnable 0 =ADCclosure (no current consumption)	ADCON0[0]		RW−0

name	state		register	address	reset value
DIVS	ADCDivided clock source selection	1 = LIRC 0 =SysClk	ADCON1[7]		RW−0
ADCS	ADCConvert Clock Source DIVS = 0 / DIVS = 1 000 =SysClk/2 / 000 =LIRC/2 001 = SysClk/8 / 001 = LIRC/8 010 = SysClk/32 / 010 = LIRC/32 100 = SysClk/4 / 100 = LIRC/4 101 = SysClk/16 / 101 = LIRC/16 110 = SysClk/64 / 110 = LIRC/64 x11 = LIRC / x11 = LIRC Note:LIRC = 32kHzor256kHz,depending onLFMODthe value of		ADCON1[6:4]	0x9F	RW−000
ANSEL	Pin Analog Select Bits 1 =analog input 0 =numberIO Note: whenADCconfigured to sample the internal1/4 VDDchannel, you need to setANSEL[7] = 1		ANSEL[7:0]	0x91	RW−1111 1111

surface1-2 ADCRelated user registers

name	address	bit 7	bit 6	bit 5	bit 4	bit 3	Bit 2	bit 1	bit 0	reset value
ADRESL	0x9E	A/DConversion Result Least Significant Bit								xxxx xxxx
ADRESH	0x1E	A/DConversion result high significant bit								xxxx xxxx
ADCON0	0x1F	ADFM	VCFG<1:0>		CHS<2:0>			GO/DONE	ADON	0000 0000
ADCON1	0x9F	DIVS	ADCS<2:0>			–				0000----

surface1-3ADCRelated user register address

1.2. ADCsconfiguration

configurationADCIncluding the following settings (need to be set when changing the configurationADON=0to closeA/Dconversion or external trigger):

- channel selection
- ADCreference voltage
- ADCConvert Clock Source
- Convert result format
- trigger source
- Response (interrupt setting)

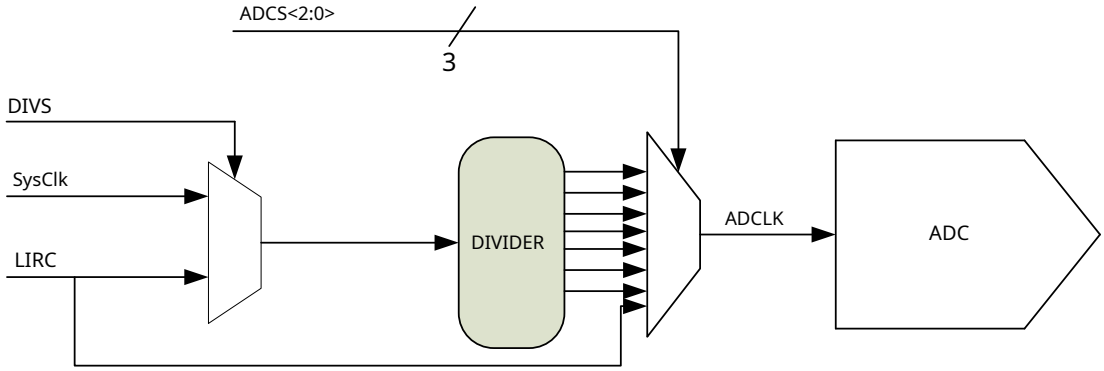
channel selection–CHSregister determines which channel is connected to theADCConverted sample-and-hold circuit. correspondingI/Oneed to be set by TRISx = 1andANSELx = 1to configure as an analog input.

ADCReference voltage ($V_{ADC-REF}$)–ADCTo measure an input analog voltage relative to a reference voltage:V_{REF}. The negative reference voltage is always referenced to ground, the positive reference voltage can be chosen as:

- VREFoptionalVDD
- internal reference voltage2V
- internal reference voltage3V
- External reference voltage (VREFforPC0)

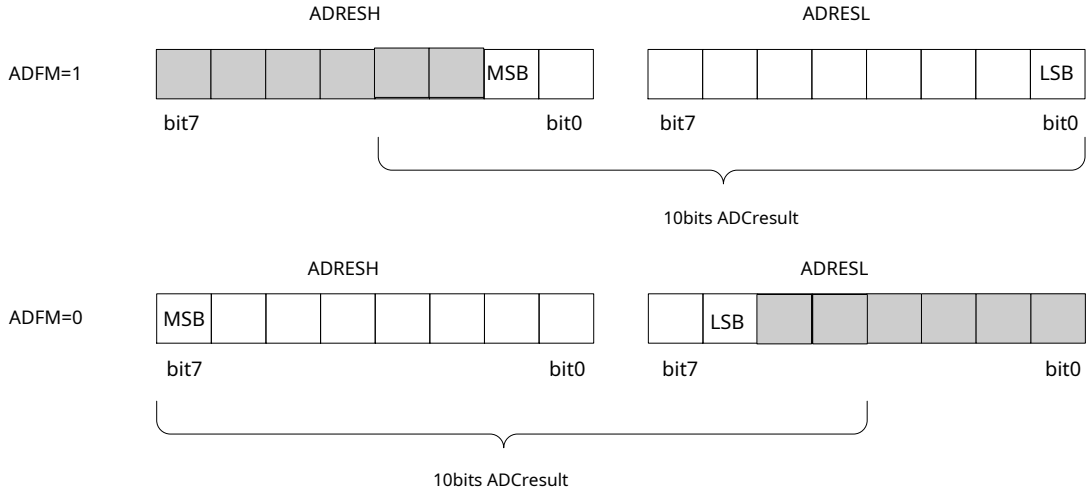
ADCConversion Clock Selection–ADCCan be selected by command13clock frequency (see "ADCS",**surface1-2**):

- DIVS = 0whenSysClk/NorLIRC;DIVS = 1whenLIRC/N;N = 1, 2, 4, 8, 16, 32, 64 LIRC (256
- kHzor32kHz,See "LFMOD",Error! Reference source not found.)



picture1-2ADCclock configuration

Convert result format–A/DThe conversion result can be stored in left-aligned or right-aligned formats (see "ADFM",**surface1-2**).



picture1-3 ADCConvert result format

1.2.1. ADCstrigger

ADCConversion can be done by the instruction (GO/DONE=1),ECCP1Special event trigger fires.GO/DONEStarted immediately after being asserted by the instruction A/Dconvert. whenECCP1Select as special event trigger mode,TMR1H/LandCCPR1H/LWhen a match occurs in the register, a special event trigger output is generated to start aADCconvert.

Note: ADCNew trigger conditions are ignored until the transition is complete.

1.2.2. ADCsabort conversion

sometimes need to abortADCconversion, such as starting a newADCwhen sampling.

- Can be set by softwareGO/DONE = 0to stopADC. When selecting a special event trigger, it
- must be closed byADCmodule(ADON = 0)to stopADC.
- whenADCconversion is aborted when theADRESHandADRESLWill not be updated, but keep the previous conversion result value. When the
- system is reset, since the corresponding registers are reset, soADCwill be discontinued, andADCModule is closed.

1.2.3.to interrupt

ADCThe module will set the corresponding interrupt flag bit when the following events occur:

- ADCconversion complete (ADCIF)

The interrupt module has its corresponding interrupt enable bit (ADCIE), and higher-level peripheral total interrupts (PEIE), and the highest level of global interrupt (GIE).

Regardless of whether the interrupt enable bit is enabled or not, the corresponding interrupt flag will be set when an interrupt event occurs. Whether to triggerto interrupt and / or wake up from sleep depends on the corresponding enable control bit (GIE, PEIE, ADCIE).

1.3. ADCssampling time

The sample time, the sample-and-hold time, must be long enough to ensure that the internalADCvoltage regulation at the input channel voltage of the0.01%within the error to achieve 10bitThe accuracy of (0.097%). The relationship between the sampling time and the external series resistance is as follows (**surface1-4**):

$$T_{ACQ} > 0.16 \times R\ (\mu s);\ R\text{The unit isk-.}$$

When sampling timeT_{ACQ}for2μs, the external series resistor must≤ 12.5k-. If a larger series resistor is used, theT_{ACQ}will increase proportionally. Junction leakage currents limit the maximum series resistor value allowed. for5nAThe junction leakage current at50k-The series resistance of the will produce0.25mV (2Vreference voltage0.0125%) pressure drop. And when the temperature exceeds100°C, the junction leakage current will increase significantly. Therefore, the smaller the series resistance, the better. -

Series resistance value	T_{ACQ}
> 50k-----	(Not recommended)
50k-----	≥ 8.0μs
25k-----	≥ 4.0μs
12k-----	≥ 2.0μs
< 12 k-----	≥ 2.0μs

surface1-4 different external series resistors with the shortestT_{ACQ}Correspondence

The sample-and-hold time is the internalADCTime to observe the voltage of the input channel.

Start of sample and hold time = after channel switching orADCAfter stabilization, whichever is later.

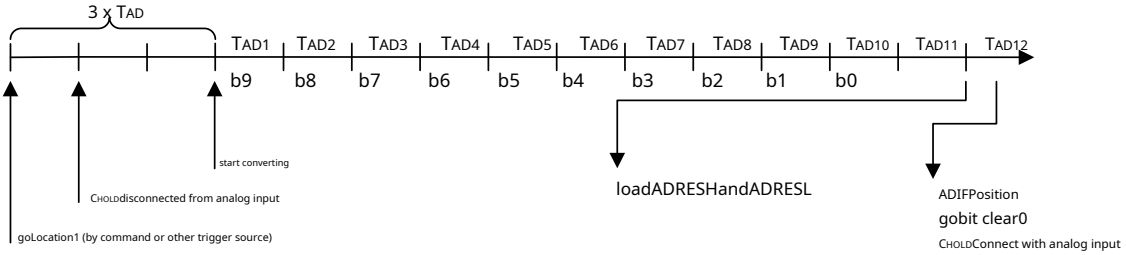
end of sample hold time = end of delay0arrive1indivualT_{AD}in time.

1.4. ADCsMinimum sampling time

T_{AD}forADCthe clock cycle. convert10bit (bit)need11.5T_{AD}, synchronization requires2 – 3 x T_{AD},complete10-bitMinimum time required for conversion:

$$T_{ACQ} + (2 + 11.5) \times T_{AD} = T_{ACQ} + 13.5\ T_{AD}$$

Guaranteed to be true10-bitThe highest conversion sample rate for accuracy is approximately34 kHz (or ~29 μs/sampling).



picture1-4 Analog-to-digital conversionT_{AD}cycle

1.5. ADCsExample conversion steps

set upADC:

1.Configure port:

a.set upTRISx = 1, disables pin output drive

b.set upANSELx = 1, turn off digital input, weak pull-up and weak pull-down functions

2.configurationADCmodule:

a.chooseADCConvert Clock Source

b.chooseADCreference voltage

c.chooseADCTriggering conditions:GO/DONEorCCP1special event trigger

d.Select conversion result format

3.configurationADCinterrupt (optional):

a.EnableADCto interrupt

b.Enable total peripheral interrupt

c.Disable the global interrupt (enable it if the interrupt service routine needs to be executed)

4.OpenADCmodule. then wait for the requiredADCstable scheduleT_{ST}(~15μs),when$V_{ADC-REF}$When selecting the internal reference voltage, you need to wait for the stabilization time of the internal reference voltageT_{VRINT}(see"T_{VRINT}",**chapter**Error! Reference source not found.)andT_{ST} whichever is longer, themax(T_{VRINT},T_{ST}).

So far,ADCDifferent channels are ready to be sampled. When sampling an input channel:

1. ADCsThe input is selected as the channel to be measured (see "CHS").

2.Clear if necessaryADCConversion complete interrupt flag.

3.pair sampling timeT_{ACQ}There are minimum requirements,T_{ACQ}be long enough to ensure that the internalADCThe input capacitor is fully charged to the input channel voltage of the 0.01% within error. In addition, depending on the trigger type, after switching channels orADCAfter stabilization (whichever is later) there may be a certain delay before triggering.

a.For software triggering, an additionalT_{ACQ}time.

5.Set by the instruction after waiting the required delayGO/DONE, or wait forCCP1Special event triggers automatic setGO/DONE, to startA/Dconvert.

6.wait byADCConversion done:

a.InquireGO/DONEbit

b.waitADCInterrupt (when interrupt is enabled)

7.readADCconversion result

8.Clear if necessaryADCConversion complete interrupt flag.

Note:

1.AlthoughGO/DONEandADONin the same register (ADCON0), but should not be set at the same time.

2. ADCsThe configuration cannot be changed during conversion or while waiting for a special event to trigger. recommended inADON = 0when making changes.

The following isADCProgram example (input sampling channel isPA0,ADCthe clock isLIRC):

```
BANKSEL ADCON1
LDWI B'00110000'          ; ADC LIRC clock
STR ADCON1

BANKSEL TRISA
BSR TRISA, 0              ; Set PA0 to input
BANKSEL ANSEL
BSR ANSEL, 0              ; Set PA0 to analog

BANKSEL ADCON0
LDWI B'10000001'          ; Right justify,
STR ADCON0                ; VDD, Vref, AN0, On

LCALL StableTime          ; ADC stable time
LCALL SampleTime          ; Acquisition delay, TACQ

BSR ADCON0, GO            ; Start conversion
BTSC ADCON0, GO           ; Conversion done?
LJUMP $-1                 ; No, test again

BANKSEL ADRESH;
LDR ADRESH, W             ; Read upper 2 bits ;
STR RESULTHI              Store in GPR space
BANKSEL ADRESL;
LDR ADRESL, W             ; Read lower 8 bits ;
STR RESULTLO              Store in GPR space
```

2.Application example

```
/*file name:TEST_61F02x_ADC.C
* Function:     FT61F02x-ADCDemo
*IC:            FT61F023 SOP16
* Crystal:      16M/2T
* illustrate:   Procedure according toAN2(PA2)port voltage to adjust thePWM3P (PC4)duty cycle AN2The
*               higher the pin voltage (0-VDD) PWM3 (PC2)The larger the duty cycle (2K,0-99%)
*
*                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|-------AN2
* NC----------|5(PC3)      (PA3)12|---------NC
* NC----------|6(PC2)      (PC0)11|---------NC
* NC----------|7(PA4)      (PC1)10|---------NC
* NC----------|8(PC5)      (PC4)09|----PWM3
*                 ------------------------
*/
# include "SYSCFG.h"
//*************************Macro definition ***************************
#define    unchar        unsigned char
#define    unint          unsigned int

//PWMPin input and output control
#define PWM3Dir TRISC4

volatile unint            TestADC;
/*-------------------------------------------------- -
* Function name:POWER_INITIAL
* Function:    Power-on system initialization
* enter:    none
* output:   none
 ------------------------------------------------
--------- */ void POWER_INITIAL (void)
{
    OSCCON = 0B01110001;            //IRCF=111=16MHz/2T=8MHz,0.125μs//Temporarily
    INTCON = 0;                     disable all interrupts

    PORTA = 0B00000000;
    TRISA = 0B00000100;             //PAinput Output1-enter0-output
                                    //PA2-enter
    PORTC = 0B00000000;
```

```
    TRISC = 0B00000000;              //PCinput Output1-enter0-
    WPUA = 0;                        output //ban allPApull up
    WPUC = 0;                        //ban allPCpull up

    OPTION = 0B00001000;             //Bit3=1, WDT MODE, PS=000=WDT RATE 1:1
    MSCKCON = 0B00000000;
    //Bit6:    0-prohibitPA4,PC5Regulated
    //Bit5:    output 0-TIMER2the clock isFosc
    //Bit4:    0-prohibitLVR
    CMCON0 = 0B00000111;             //turn off the comparator,Cxfor numbersIOmouth
}
/*---------------------------------------------- -
* Function name:Delay Us
* Function:    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:ADC_INITIAL
* Function:    ADCinitialization
* enter:       none
* output:      none
  -------------------------------------------------
----- */ void ADC_INITIAL (void)
{
    ADCON1 = 0B01100000;
    //Bit7:        DIVS=0,clock selectionFOSC
    //Bit[6:4]:    ADCS[2:0]=110,frequency divisionFOSC/64

    ADCON0 = 0B10001001;
    //Bit7: ADFM=1,Results right aligned
    //Bit[6:5]: VCFGreference voltage
    //           00-reference voltageVDD
    //           01-reference voltage internal2V
    //           10-reference voltage internal3V
    //           11-reference voltageVref //Bit[4:2]:CHS=010-chooseAN2aisle
```

```
    //Bit1:     GO,ADtransition status bit
    //Bit0:     ADON=1,ADCEnable
    ANSEL = 0B00000100;                 //EnableAN2for analog input
}
/*---------------------------------------------------- -
* Function name:GET_ADC_DATA
* Function:     read channelADCvalue
* enter:        AN_CNchannel number
* output:       INTtypeADValue (single sample without filtering)
  ---------------------------------------------
------------ */ unint GET_ADC_DATA(unchar AN_CH) {

    unchar    i;
    unchar    ADCON0Buff;
    unint     tBuffer = 0;
    unint     ADC_DATA=0;

    ADCON0Buff = ADCON0 & 0B11100011; //clear channel value

    AN_CH <<=2;
    ADCON0Buff |= AN_CH;                //Bit[4:2]:CHS=010,chooseAN2aisle

    ADCON0 = ADCON0Buff;                //reload channel value
    DelayUs(20);
    GO_DONE = 1;                        //start upADC
    while( GO_DONE==1 );                //waitADCconversion complete

    ADC_DATA = ADRESH;
    ADC_DATA <<= 8;
    ADC_DATA |= ADRESL;                 //10Bit ADCvalue integration
    tBuffer = ADC_DATA;
    return tBuffer;
}
/*---------------------------------------------------- -
* Function name:    PWM_INITIAL
* Function:         PWMinitialization
* set period =2 TMRxPS*2 NBit*TPxCK
*           = 2 0*2 8*[(T3CKDIV+1)/16000000] =
*           1*256*[(30+1)/16000000] = 0.496ms
* set pulse width =2TMRxPS*(PRx)*TPxCK
*           = 2 0*128*[(T3CKDIV+1)/16000000] =
*           1*128*[(30+1)/16000000] = 0.248ms
  ---------------------------------------------------
----- */ void PWM_INITIAL (void)
```

```
{
    PWM3Dir = 1;                        //PWM3outputPINTemporarily in input mode

    PWM3CR0 = 0B00110010;
    //Bit7:     interrupt select bit0-Generate interrupt on counter overflow
    //Bit[6:4]:Period bit selection011-8bit //
    Bit[3:1]:clock selection      001-internalRCfast clock/(T3CKDIV+1)
    //Bit0: PWM/BUZZERchoose0-PWMoutput

    PWM3CR1 = 0B10000000;
    //Bit7:       1-TMR3forPWM/BUZZERmodel
    //Bit6:       0-PWM3active high
    //Bit[5:3]:   000-PWM3The prescaler is set to1:1
    //Bit2:       0-temporarily closedTMR3
    //Bit1:       0-prohibitTMR3to interrupt
    //Bit0:       0-TMR3Interrupt flag bit read only
    TMR3H = 0;
    T3CKDIV = 30;
    PR3L = 128;                         //duty cycle50%
}
/*---------------------------------------------- -
* Function name:main
* Function:   main function
* enter:    none
* output:   none
  ------------------------------------------------------------
*/ void main()
{
    POWER_INITIAL();                    //system initialization
    ADC_INITIAL();                      //ADCinitialization
    PWM_INITIAL();                      //PWMinitialization
    TMR3ON = 1;                         //T3turn onPWM3
    PWM3Dir = 0;                        //PWM3 PINSet to output mode to allowPWMoutput
    while(1)
    {
        TestADC = GET_ADC_DATA(2);      //aisle2ADvalue
        PR3L = TestADC >> 2; //WillADvalue assigned toPWMofPR3LadjustPWM3Poutput duty cycle NOP();

        NOP();
    }
}
```

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

* Information furnished is believed to be accurate and reliable. However, Fremont Micro Devices Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents of other rights of third parties, which may result from its use. No license is granted by implication or otherwise under any patent rights of Fremont Micro Devices Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. components in life support devices or systems without express written approval of Fremont Micro Devices Corporation. The FMD logo is a registered trademark of Fremont Micro Devices Corporation.All other names are the property of their respective owners.