HY16F18X
HYCON IP User's Manual

Table of Contents

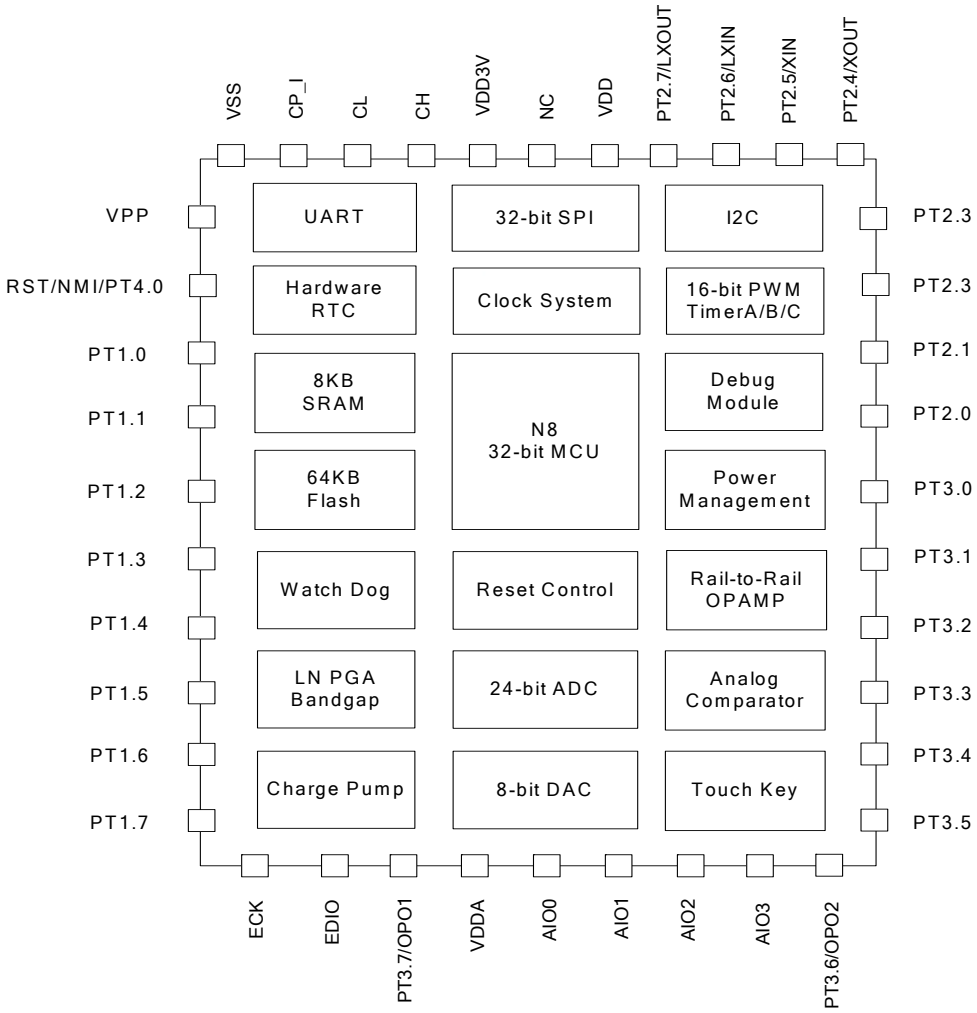
1. Document Description

HYCON IP(Intellectual Property) represents all internal IP of HYCON 32-bit MCU. This document aims at describing SOC IC internal digital, analog, communication and peripheral IP of HY16F18X Series, of which can be grouped into four categories:

(1) Digial IP：TimerA/TimerB/TimerC/WDT/PWM/Hardware RTC
(2) Analog IP：DAC/ADC/OPAMP/Analog CMP
(3) Communication IP：Hardware 32-bit SPI/ Hardware UART/ Hardware I2C
(4) Peripheral IP：GPIO/Sleep/Idle

2. IC Description

Basic description of each HY16F188 IP.



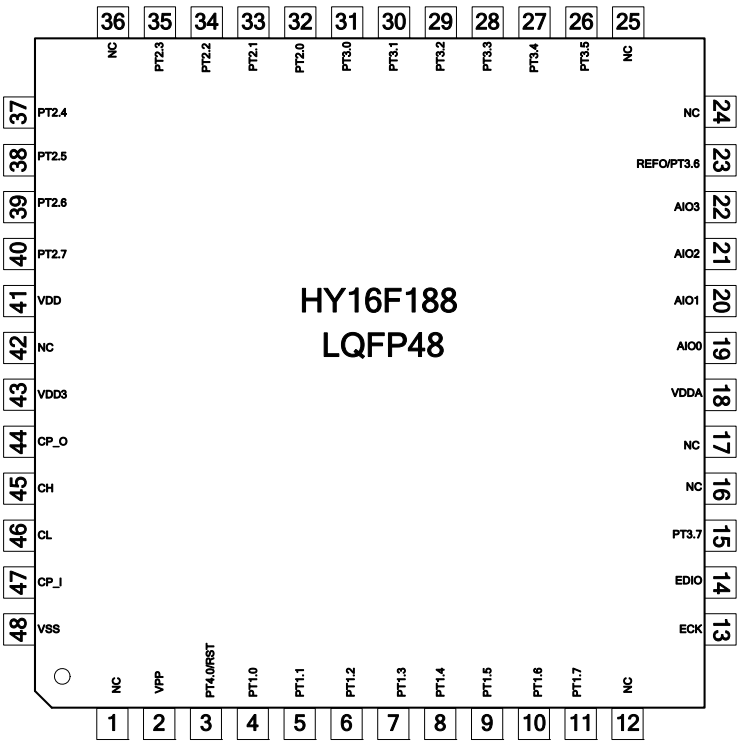
(01) Adopting Andes Technology newest 32-bit CPU core, N801 processor.
(02) Voltage operation range: 2.0~3.6V; temperature operation range: -40℃~85℃.
(03) Support external 20MHz crystal oscillator or internal 20MHz RC oscillator, having multiple switch options of CPU operation clock source to enable the best power-efficient plan.
  (3.1) operation mode: 350uA@2MHz/2
  (3.2) standby mode: 10uA@35KHz/2
  (3.3) sleep mode: 2.5uA
(04) Program memory: 64K-Byte Flash ROM
(05) Data memory: 08K-Byte SRAM
(06) BOR and WDT function to prevent CPU from crashing
(07) 24-bit high resolution ΣΔADC
  (7.1) Built-in PGA (Programmable Gain Amplifier), 128 times max.
  (7.2) Built-in temperature sensor, TPS
(08) Built-in 1 OPA and 1 analog comparator (16 steps programmable resistor)
(09) Built-in hardware 8-bit DAC
(10) 16-bit Timer A
(11) 16-bit Timer B module has PWM waveform generating function
(12) 16-bit Timer C module has digital Capture/Compare function
(13) Hardware serial communication 32-bit SPI/I2C/UART module
(14) Hardware RTC clock function module
(15) Hardware Touch KEY function module

※ Interrupt Vice Program Declaration

No.	Interrupt Vice Program Declaration	Description	Remark
HW0	void HW0_ISR(void)	Communication IP	UART/I2C/32-bit SPI
HW1	void HW1_ISR(void)	Digital IP	Timer A/B/C & WDT/RTC
HW2	void HW2_ISR(void)	ADC IP	SD 24-bit ADC
HW3	void HW3_ISR(void)	CMP IP	Analog CMP/OPAMP
HW4	void HW4_ISR(void)	PT1 IP	PT1.0~PT1.7
HW5	void HW5_ISR(void)	PT2 IP	PT2.0~PT2.7

	Digital IP	Analog IP	Communication IP	Peripheral IP
01	TimerA	8-bit DAC	Hardware 32-bit SPI	GPIO
02	TimerB	R2R OPAMP	Hardware UART	Sleep Mode
03	TimerC	24-bit SD ADC	Hardware I2C	Idle Mode
04	WDT	Analog CMP	–	–
05	PWM	–	–	–
06	Hardware RTC	–	–	–

GPIO Port	OSC	Interrupt	Timer C	SPI	IIC	UART	CMP	Analog	PWM
Priority	0	0	0	1	2	3	4	5	6
PT1.0		INT1.0	TCI1_1	CS_1	SCL_1	TX_1	CH1		PWM0_1
PT1.1		INT1.1	TCI2_1	CK_1	SDA_1	RX_1	CH2		PWM1_1
PT1.2		INT1.2	TCI1_2	MISO_1	SCL_2	TX_2	CH3		PWM0_2
PT1.3		INT1.3	TCI2_2	MOSI_1	SDA_2	RX_2	CL1		PWM1_2
PT1.4		INT1.4	TCI1_3	CS_2	SCL_3	TX_3	CL2		PWM0_3
PT1.5		INT1.5	TCI2_3	CK_2	SDA_3	RX_3	CL3		PWM1_3
PT1.6		INT1.6	TCI1_4	MISO_2	SCL_4	TX_4	CL4		PWM0_4
PT1.7		INT1.7	TCI2_4	MOSI_2	SDA_4	RX_4	CMPO1		PWM1_4
PT2.0		INT2.0	TCI1_5	CS_3	SCL_5	TX_5			PWM0_5
PT2.1		INT2.1	TCI2_5	CK_3	SDA_5	RX_5			PWM1_5
PT2.2		INT2.2	TCI1_6	MISO_3	SCL_6	TX_6			PWM0_6
PT2.3		INT2.3	TCI2_6	MOSI_3	SDA_6	RX_6			PWM1_6
PT2.4	LSXT1	INT2.4	TCI1_7	CS_4	SCL_7	TX_7			PWM0_7
PT2.5	LSXT2	INT2.5	TCI2_7	CK_4	SDA_7	RX_7			PWM1_7
PT2.6	HSXT1	INT2.6	TCI1_8	MISO_4	SCL_8	TX_8			PWM0_8
PT2.7	HSXT2	INT2.7	TCI2_8	MOSI_4	SDA_8	RX_8			PWM1_8
PT3.0							OPO1		
PT3.1							OPO2	DAO	
PT3.2								AIO4	
PT3.3								AIO5	
PT3.4								AIO6	
PT3.5								AIO7	
PT3.6								REFO	
PT3.7								OPO	
PT4.0		RST/NMI							
AIO0								AIO0	
AIO1								AIO1	
AIO2								AIO2	
AIO3								AIO3	



3. Digital IP (Timer A)

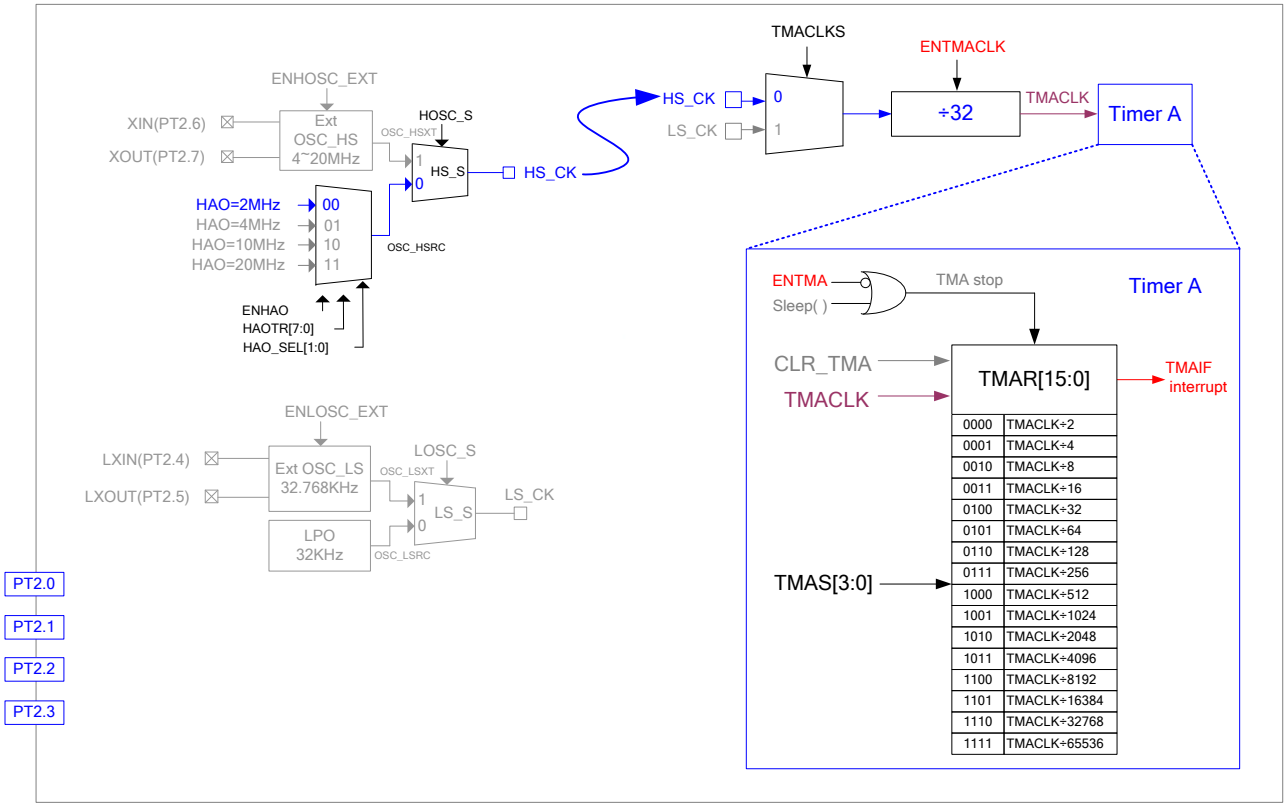
3.1 Example Name

Timer A usage and description

3.2 Example Description

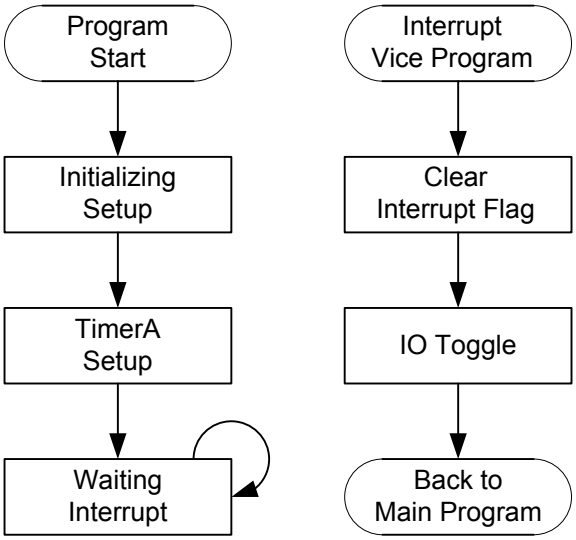
(1) Using TimerA to interrupt
(2) Every time TimerA interrupted, IO toggled.
Ex: Before interrupt: 0X5, after interrupt: 0XA

3.3 System Setup



(1) Default operation frequency of clock source is 2MHz (high speed internal oscillator), up to 20MHz@2.6V max..
(2) Adopting HYCON C library, DrvTMA_Open(X,Y) can start TimerA IP and Timer A Clock.
X represents TMAR frequency divider, select 15 is divided by 65536.
Y represents selecting high speed or low speed CPU Clock to enter Timer A IP
(3) HYCON C library DrvTIMER_EnableInt(E_TMA), meaning to enable Timer A interrupt.
(4) HYCON C library DrvTIMER_ClearIntFlag(E_TMA), meaning to clear Timer A interrupt flag.
(5) Timer A/B/C and WDT is classified as HW1 interrupt, using form is void HW1_ISR(void).

3.4 Software Flow



3.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  unsigned int i;
04
05  int main(void)
06  {
07          i=0X05;
08          DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);   //PT2_0~3 Set Output
09          DrvGPIO_SetPortBits(E_PT2,i);           //PT2 Output i=0X05
10
11          DrvTMA_Open(15,0);                      //Timer A Overflow
12                                                  //15 : taclk/65536/32;TMRDV=÷32
13                                                  // 0 : HS_CK
14
15          DrvTIMER_ClearIntFlag(E_TMA);           //Clear Timer A interrupt flag
16          DrvTIMER_EnableInt(E_TMA);              //Timer A interrupt enable
17
18          SYS_EnableGIE(7);                       //Enable GIE
20
21          while(1);                               //Wait for Interrupt
22  }
23
24  void HW1_ISR(void)
25  {
26          DrvTIMER_ClearIntFlag(E_TMA);           //Clear TMA interrupt flag
27          i=i^0XF;                                //i XOR 0XF
28          DrvGPIO_SetPortBits(E_PT2,i);           //PT2 Output i=0X0A~0X05
29  }
30
```

4. Digital IP (Timer B)

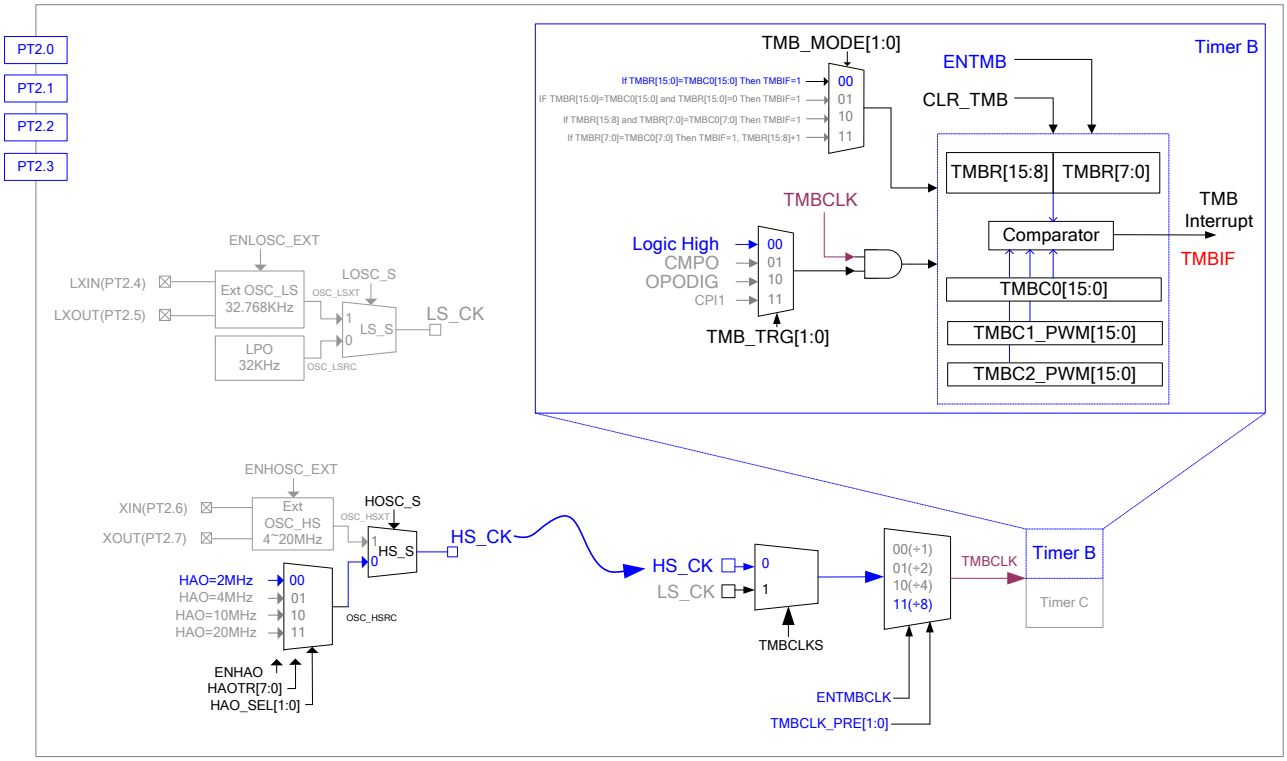
4.1 Example Name

Timer B usage and description

4.2 Example Description

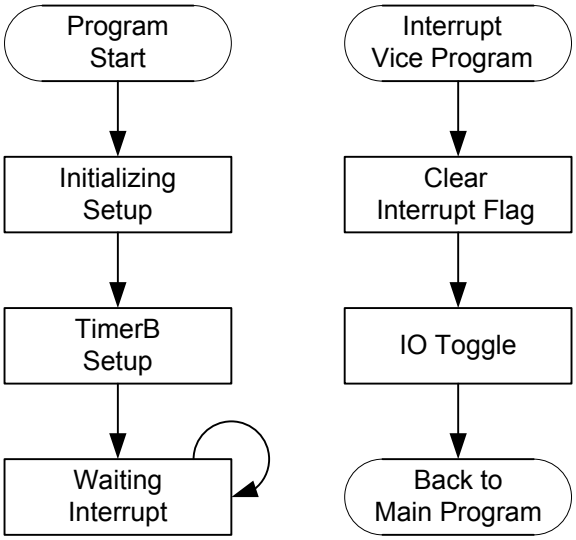
(1) Using TimerB interrupt
(2) Every time TimerB interrupted, IO toggled.
Ex. Before interrupt: 0X5, after interrupt: 0XA

4.3 System Setup



(1) Adopting HYCON C library, DrvTMBC_Clk_Source(X,Y) can select Timer B clock source. X is to select high/low speed. X=0 is HS_CK, Y is to select frequency divider. If Y=3, it means Timer B IP clock source frequency divides 8.
(2) Library DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF): TimerB IP setup including Mode select, enabling Timer B and Timer B counter setup.
(3) HYCON C library DrvTIMER_EnableInt(E_TMB): to enableTimer B interrupt.
(4) HYCON C library DrvTIMER_ClearIntFlag(E_TMB): to clear TimerB interrupt flag.
(5) Timer A/B/C and WDT is classified as HW1interrupt, using form is void HW1_ISR(void).

4.4 Software Flow



4.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  unsigned int i;
04
05  int main(void)
06  {
07          i=0X05;
08          DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);          //PT2_0~3 Set Output
09          DrvGPIO_SetPortBits(E_PT2,i);                  //PT2 Output i=0X05
10
11          DrvTMBC_Clk_Source(0,3);                       //Timer B Prescaler 1
12                                                         //0: HS_CK,clock source.
13                                                         //3: clock divider.÷8
14
15          DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF); //Timer B overflow 0XFFFF
16          DrvTIMER_ClearIntFlag(E_TMB);                  //Clear TMB interrupt flag
17          DrvTIMER_EnableInt(E_TMB);                     //Timer B interrupt enable
18
19          SYS_EnableGIE(7);                              //Enable GIE
20
21          while(1);                                      //Wait for Interrupt
22  }
23
24  void HW1_ISR(void)
25  {
26          DrvTIMER_ClearIntFlag(E_TMB);                  //Clear TMB interrupt flag
27          i=i^0XF;                                       //i XOR 0XF
28          DrvGPIO_SetPortBits(E_PT2,i);                  //PT2 Output i=0X0A~0X05
29  }
30
```

5. Digital IP(Timer C)

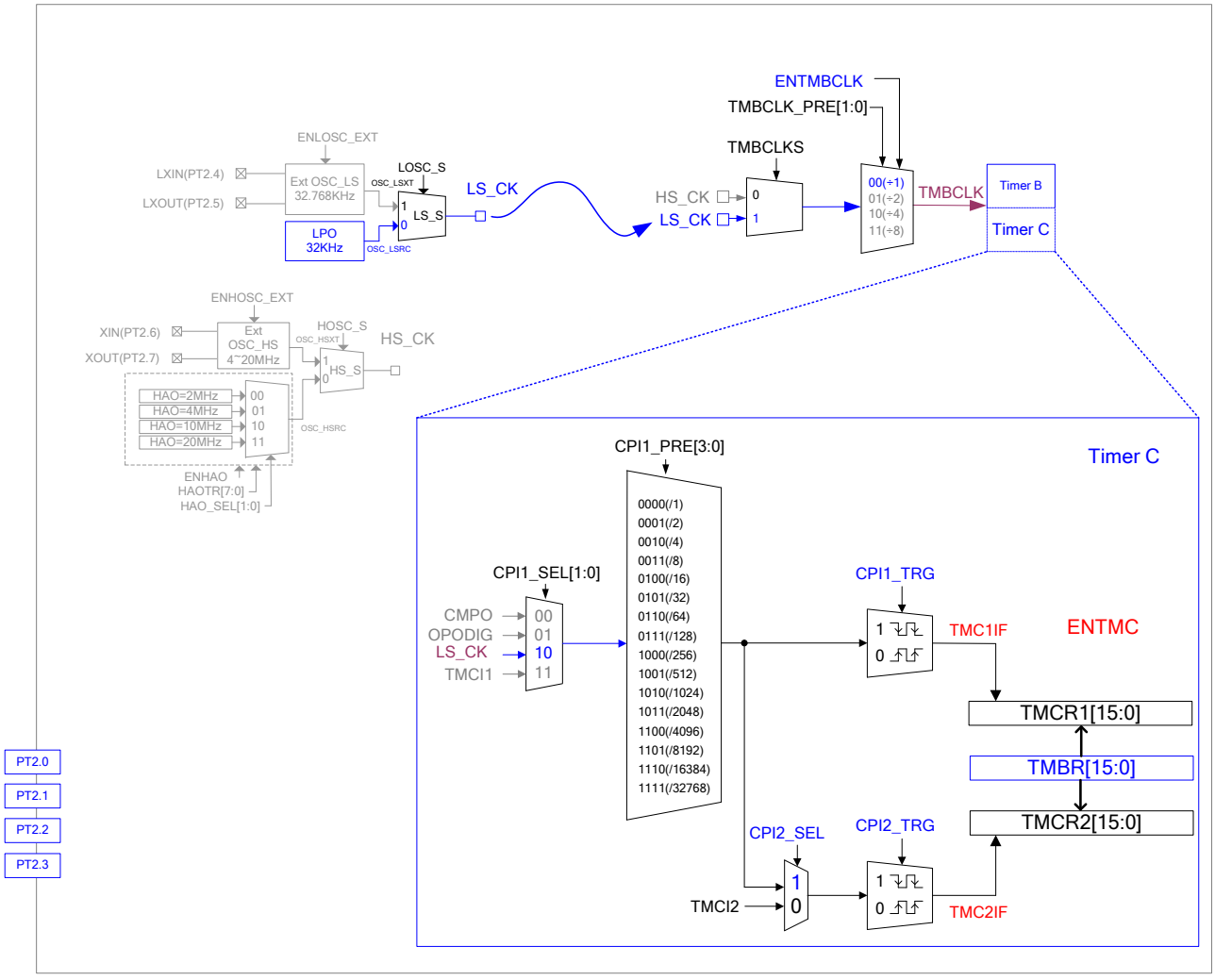
5.1Example Name

Timer C usage and description

5.2 Example Description

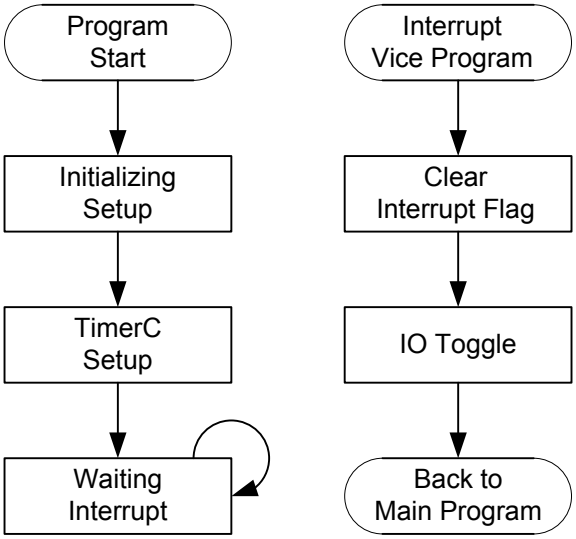
(1) Using TimerC to interrupt
(2) Every time TimerC interrupted, IO toggled.
Ex. Before interrupt: 0X5, after interrupt: 0XA.

5.3 System Setup



(1) DrvTMB_Open( ) will enable Timer B.
(2) DrvCapture1_Open( ); configuring Timer C 0.
(3) DrvCapture2_Open( ); condiguring Timer C1.

5.4 Software Flow



5.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  unsigned int i;
04
05  int main(void)
06  {
07          i=0X05;
08          DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);          //PT2_0~3 Set Output
09          DrvGPIO_SetPortBits(E_PT2,i);                  //PT2 Output i=0X05
10
11          DrvTMBC_Clk_Source(0,0);                       //Timer B Prescaler 1
12                                                         //0: HS_CK,clock source.
13                                                         //0: clock divider.÷1
14
15          DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF);  //Timer B overflow 0XFFFF
16
17          DrvCapture1_Open(2,14,1);                      //TimerC0 use as Capture 1
18                                                         //input source selection
19                                                         //2:LS_CK
20                                                         //14:÷16384 1:Positive-edge trigger
21
22          DrvCapture2_Open(1,1);                         //TimerC1 use as Capture 2
23                                                         // input source selection
24
25          DrvTIMER_ClearIntFlag(E_TMC0);                 //Clear TMC0 interrupt flag
26          DrvTIMER_ClearIntFlag(E_TMC1);                 //Clear TMC1 interrupt flag
27
28          DrvTIMER_EnableInt(E_TMC0);                    //TimerC0 interrupt enable
29          DrvTIMER_EnableInt(E_TMC1);                    //TimerC1 interrupt enable
30
31          SYS_EnableGIE(7);                              //Enable GIE
32
33          while(1);                                      //Wait for Interrupt
34  }
```

37		
38	void HW1_ISR(void)	
39	{	
40	DrvTIMER_ClearIntFlag(E_TMC0);	//Clear TMC0 interrupt flag
41	DrvTIMER_ClearIntFlag(E_TMC1);	//Clear TMC1 interrupt flag
42	i=i^0XF;	//i XOR 0XF
43	DrvGPIO_SetPortBits(E_PT2,i);	//PT2 Output i=0X0A~0X05
44	}	
45		

6. Digital IP(WDT)

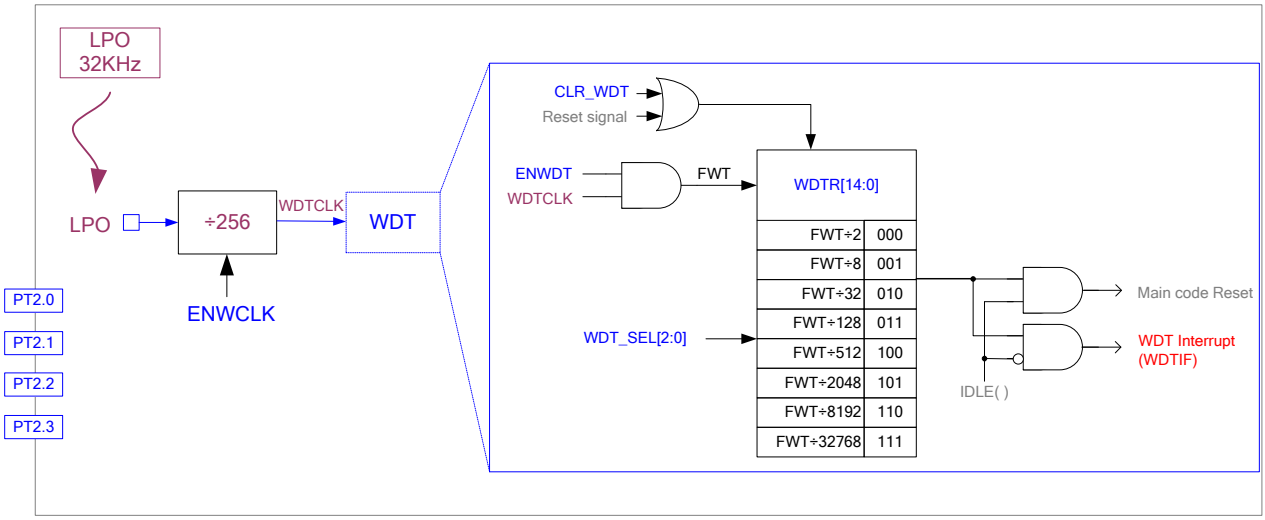
6.1 Example Name

WDT usage and description

6.2 Example Description

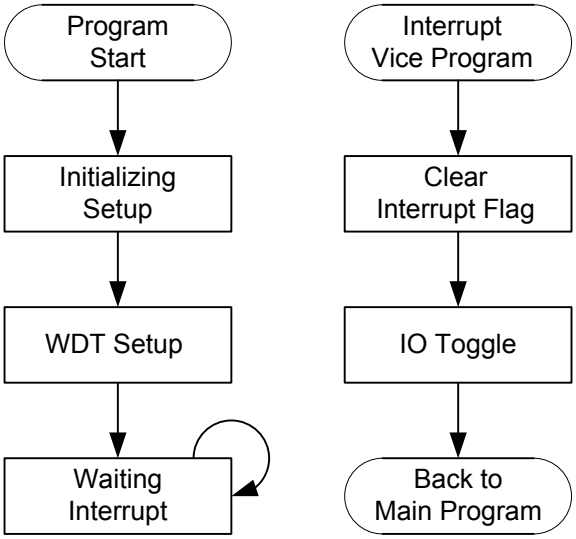
(1) Using WDT interrupt
(2) Every time WDT interrupted, IO toggled.
Ex. Before interrupt: 0X5, after interrupt: 0XA

6.3 System Setup



(1) DrvWDT_Open( ) will enable WDT
(2) DrvTIMER_EnableInt(E_WDT): enable WDT interrupt
(3) DrvTIMER_ClearIntFlag(E_WDT): clear WDT interrupt flag

6.4 Software Flow



6.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  unsigned int i;
04
05  int main(void)
06  {
07      i=0X05;
08      DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);   //PT2_0~3 Set Output
09      DrvGPIO_SetPortBits(E_PT2,i);           //PT2 Output i=0X05
10
11      DrvWDT_Open(E_IRQ,E_PRE_SCALER_D32);    //WDT IRQ open prescaler 32
12
13      DrvWDT_ClearWDT();                      //Clear WDT interrupt flag
14      DrvTIMER_EnableInt(E_WDT);              //WDT interrupt enable
15
16      SYS_EnableGIE(7);                       //Enable GIE
17
18      while(1);                               //Wait for Interrupt
20  }
21
22  void HW1_ISR(void)
23  {
24      DrvTIMER_ClearIntFlag(E_WDT);           //Clear WDT interrupt flag
25      i=i^0XF;                                //i XOR 0XF
26      DrvGPIO_SetPortBits(E_PT2,i);           //PT2 Output i=0X0A~0X05
27  }
28
```

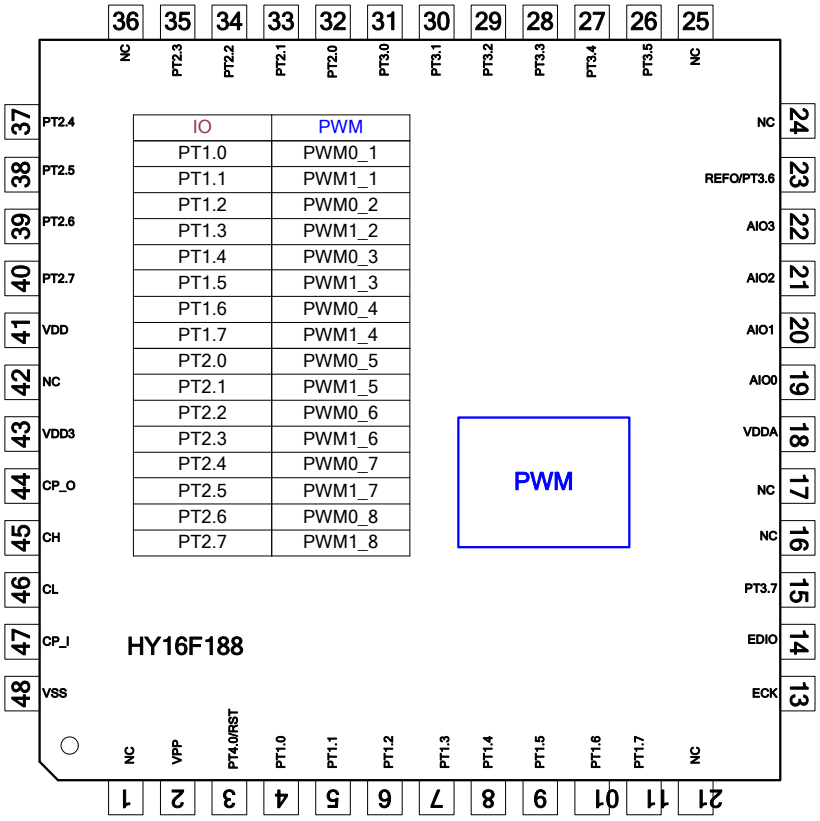
7. Digital IP (PWM)

7.1Example Name

PWM output

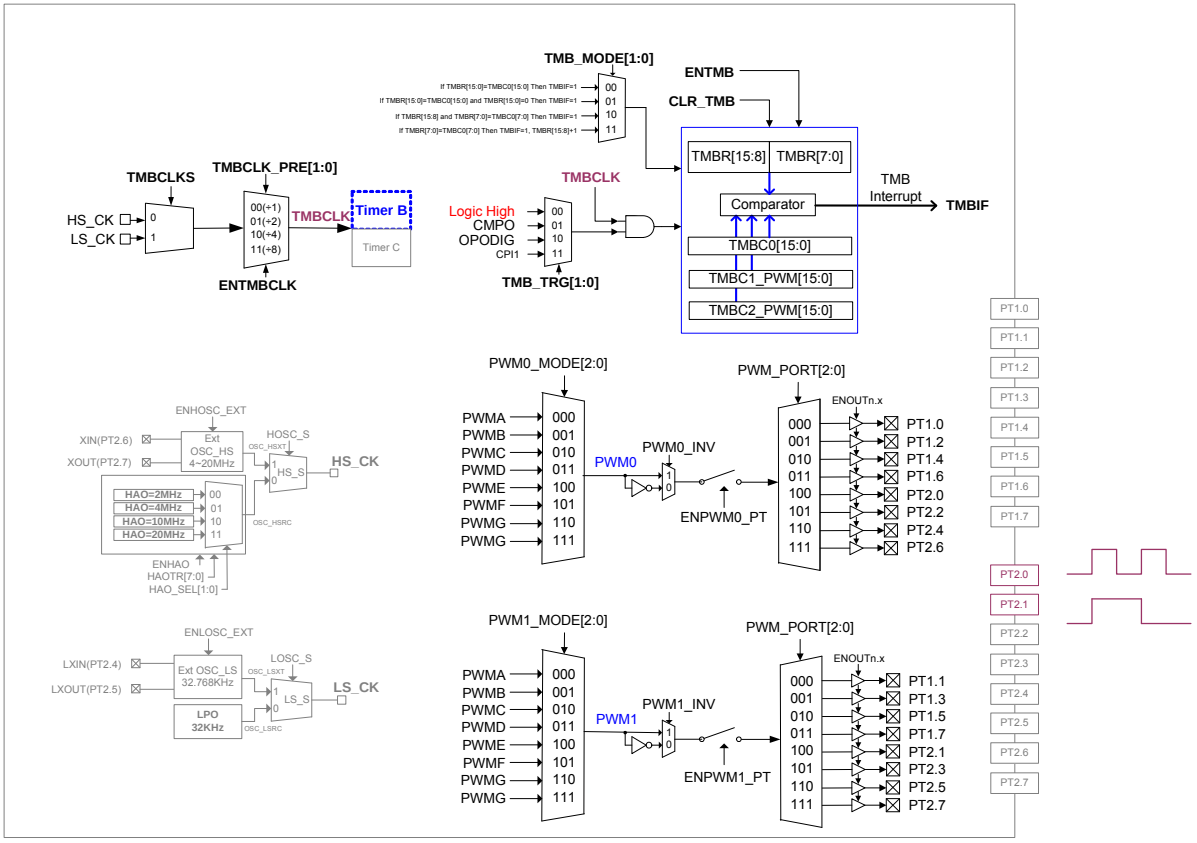
7.2 Example Description

(1) PWM register setup and IO select.
(2) PWM waveform can be observed at PT2.0 and PT2.1.

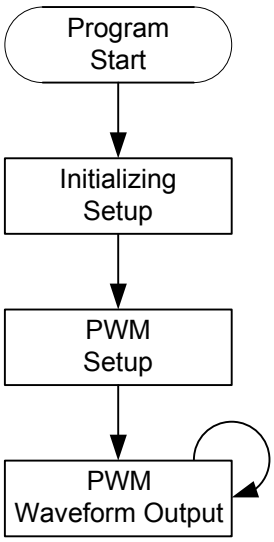


IO	PWM
PT1.0	PWM0_1
PT1.1	PWM1_1
PT1.2	PWM0_2
PT1.3	PWM1_2
PT1.4	PWM0_3
PT1.5	PWM1_3
PT1.6	PWM0_4
PT1.7	PWM1_4
PT2.0	PWM0_5
PT2.1	PWM1_5
PT2.2	PWM0_6
PT2.3	PWM1_6
PT2.4	PWM0_7
PT2.5	PWM1_7
PT2.6	PWM0_8
PT2.7	PWM1_8

7.3 System Setup



7.4 Software Flow



7.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03
04
05  int main(void)
06  {
07          SYS_DisableGIE();
08
09          DrvGPIO_Open(E_PT2,0X03,E_IO_OUTPUT);          //PT2_0~1 Set Output
10
11          DrvCLOCK_EnableHighOSC(E_EXTERNAL);            //Select HSXT 4MHz
12
13          DrvTMBC_Clk_Source(0,0);                       //TimerB Clock enable prescaler 1
14
15          DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF);  //TimerB overflow 0XFFFF
16                                                         //PWM period 0XFFFF
17
18          DrvPWM0_Open(0,1,4);                           //0:PWM mode
19                                                         //1:Normal    0:inverse
20                                                         //4:Select PT2.0 PT2.1 as PWM
21
22          DrvPWM1_Open(1,1,4);                           //1:PWM mode
23                                                         //1:Normal    0:inverse
24                                                         //4:Select PT2.0 PT2.1 as PWM
25
26          DrvPWM_CountCondition(0X7FFF,0X3FFF);          //pwm0 duty 0X7FFF
27                                                         //pwm1 duty 0X3FFF
28
29          while(1);                                      //While Loop
30  }
31
```

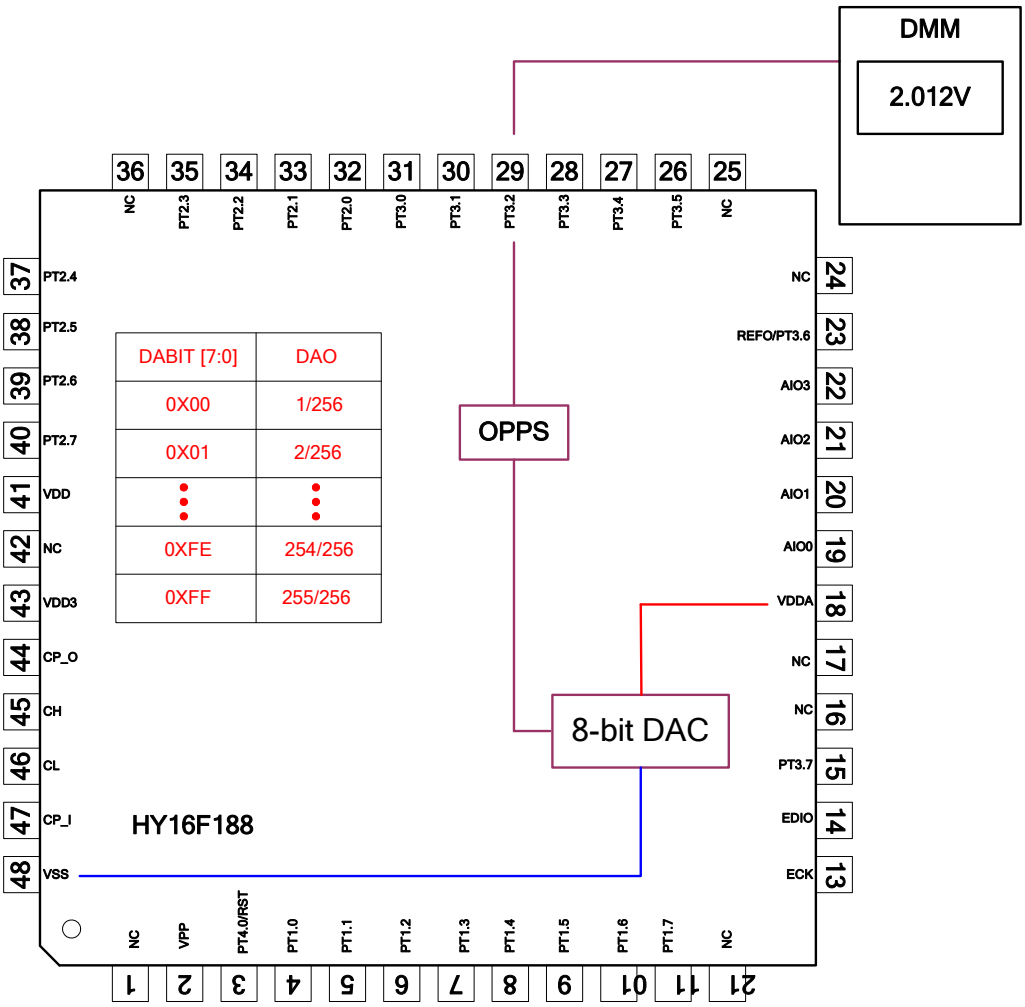
8. Analog IP(DAC)

8.1 Example Name

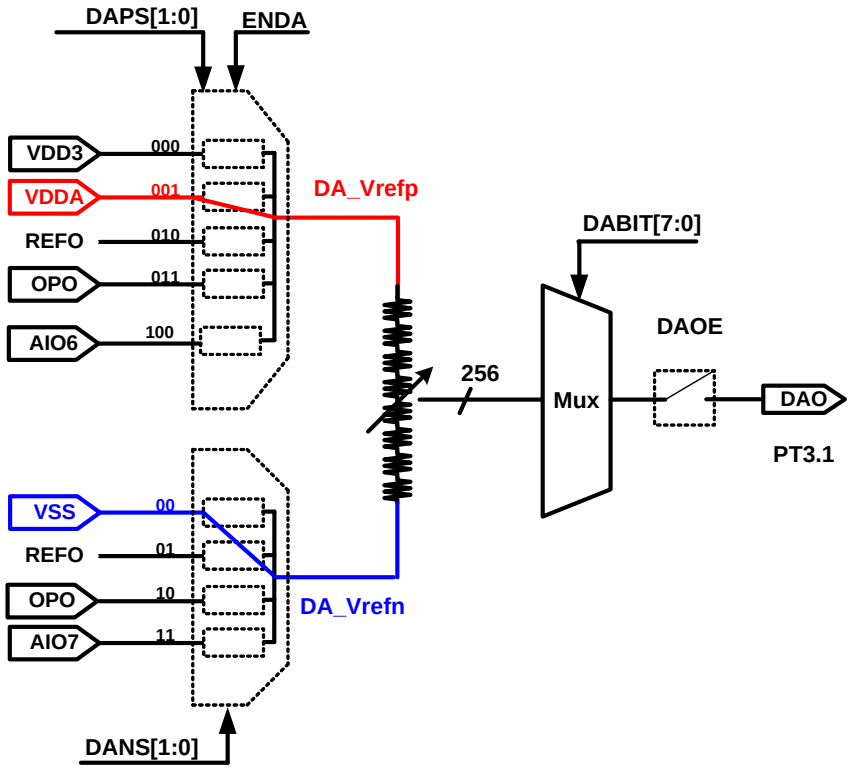
DAC usage and description

8.2 Example Description

(1) Setup analog voltage nad digital voltage
(2) Connecting REFO to DAC positive end.
(3) DAC negative end selects VSS, it will pass through AIO4 of OP (Selecting both AIO4 and DAC)
(4) It is through proper DAC network setup can AIO4 output end measures 256-tier DAC output.
(5) Or adopting DAC specialized analog votlage output pin, PT3.1.

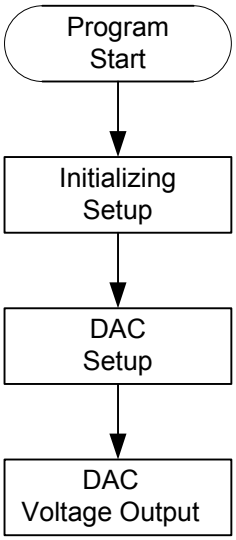


8.3 System Setup



DABIT[7:0]	DABIT
0X00	1/256
0X01	2/256
⋮	⋮
0XFE	254/256
0XFF	255/256

8.4 Software Flow



8.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  int main(void)
04  {
05      DrvPMU_VDDA_LDO_Ctrl(E_LDO);                    //LDO ON
06      DrvPMU_VDDA_Voltage(E_VDDA3_0);                 //VDDA=3.0
07
08      //DrvOP_PInput(0X06);                           //DAO output with AIO4(PIN29)
09
10      DrvDAC_Enable();                                //DAC IP enable
11      DrvDAC_EnableOutput();                          //DAC output enable
12      DrvDAC_Open(E_DAC_VDDA,E_DAC_VSSA,0X80);        //DA_Vrefp= VDDA
13                                                      //DA_Vrefn= VSS
14      dac_04=0XFF00+0X80;                             //DAC BIT 0X80
15      pio_28=0X02020000;                              //DAO output with PT3.1
16
17      while(1);                                       //while loop
18  }
19
```

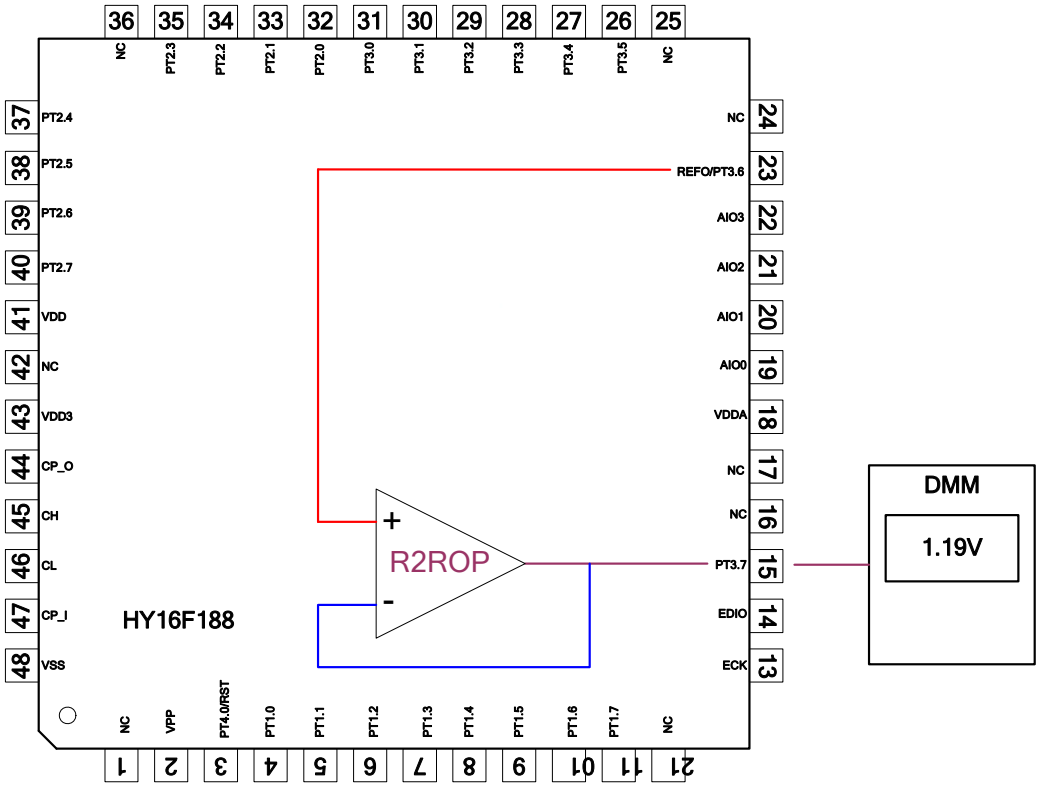
9. Analog IP (OPAMP)

9.1 Example Name

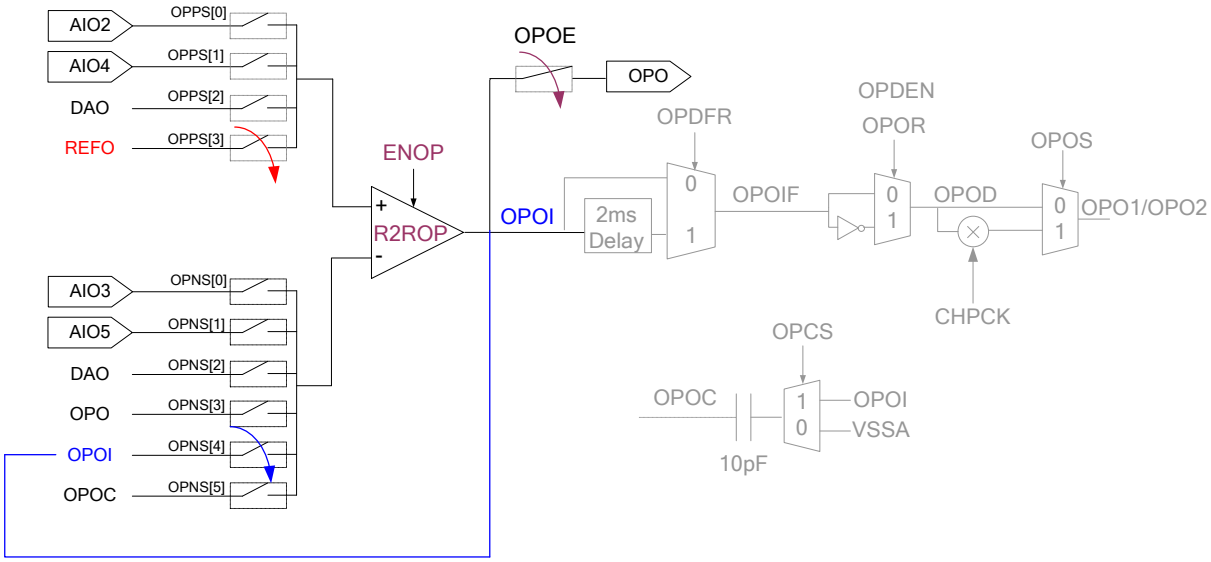
OPAMP usage and configuration

9.2 Example Description

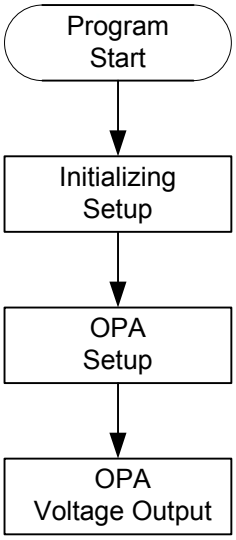
(1) Open analog voltage, REFO=1.2V
(2) Connect REFO to OPAMP positive end, V+
(3) Select negative end of OPAMP, OPOI, it will be connected to OPA Unit Gain Buffer at this time.
(4) Through proper OPAMP network setup, OPO outout end can measure REFO=1.2V.



9.3 System Setup



9.4 Software Flow



9.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  int main(void)
04  {
05      DrvPMU_VDDA_LDO_Ctrl(E_LDO);          //LDO ON
06      DrvPMU_VDDA_Voltage(E_VDDA3_0);       //VDDA=3.0
07      DrvPMU_REFO_Enable();                 //REFO ON
08
09      DrvOP_Open();
10
11      DrvOP_PInput(0X08);                   //OPA positive reference input selection REFO
12      DrvOP_NInput(0X10);                   //OPA negative reference input selection OPOI
13
14      DrvOP_OPOoutEnable();                 //OP OoutEnable
15
16      while(1);                             //while loop
17  }
18
```

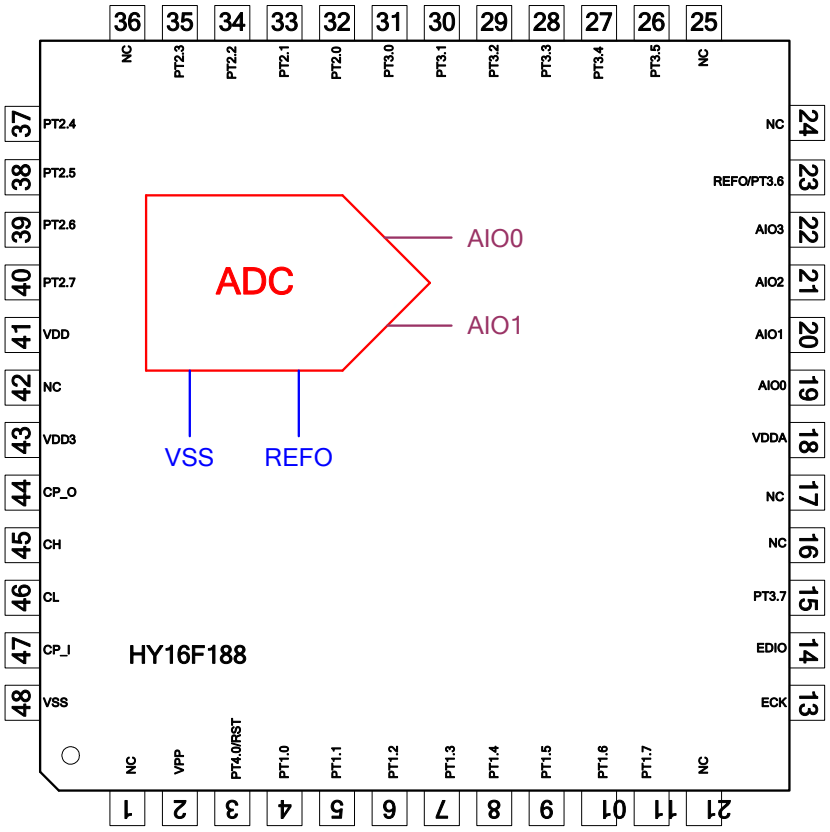
10. Analog IP (ADC)

10.1 Example Name

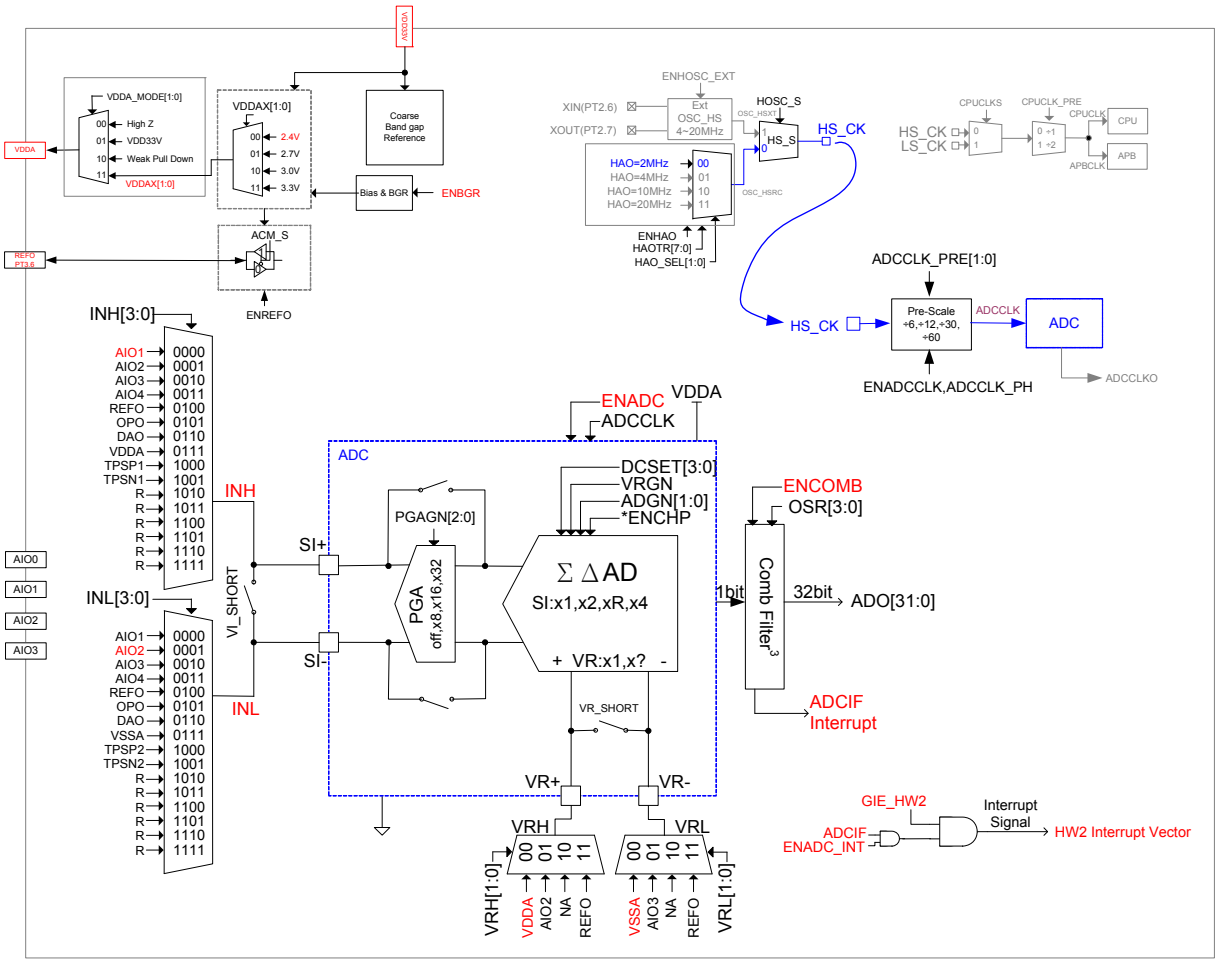
ADC interrupt usage

10.2 Example Description

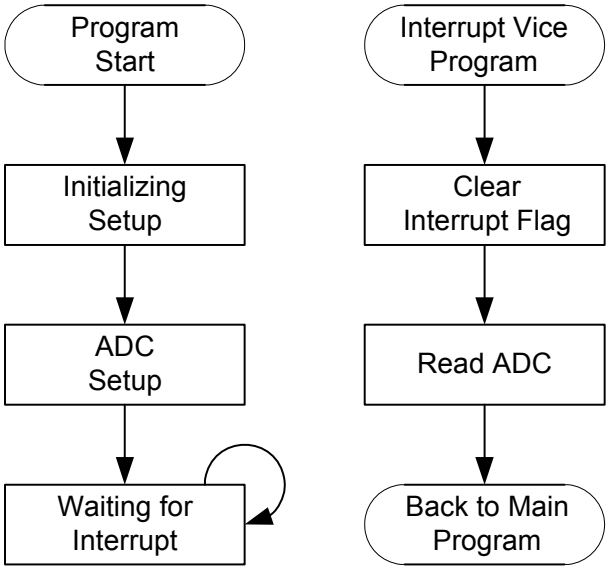
(1) It is through ADC related setup, can ADC realize interrupt function.
(2) Set up ADC power, VDDA and REFO.
(3) Set up ADC clock source, ADCCLK.
(2) ADC Vin is AIO0-AIO1.
(3) ADC Vref is REFO to VSS.



10.3 System Setup



10.4 Software Flow



10.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  unsigned int ADCData;
04
05  #define   Disable   0
06  #define   Enable    1
07
08  int main(void)
09  {
10  //Set ADC input pin
11
12    DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1); //Set ADC positive input AIO0
13                                                              //Set ADC negative input AIO1
14    DrvADC_InputSwitch(OPEN);                                 //ADC input short control
15    DrvADC_RefInputShort(OPEN);                               //ADC reference short control
16
17    DrvADC_Gain(ADC_PGA_Disable,ADC_PGA_Disable);             //Input signal gain
18
19    DrvADC_DCoffset(0);                                       //DC offset
20    DrvADC_RefVoltage(REF_BUFFER_OUT,VSSA);                   //ADC reference voltage.
21
22    DrvADC_FullRefRange(0);                                   //ADC full reference range select
23                                                              //0: Full reference range input
24                                                              //1: 1/2 reference range input
```

```
29  //ADC Speed
30
31      DrvADC_OSR(1);                          //1 : OSR=16384
32      DrvADC_CombFilter(Enable);              //Enable OSR
33
34      DrvADC_ClkEnable(0,1);                  //Setting ADC CLOCK
35                                              //ADCK=HS_CK/6
36                                              // Rising edge is high
37
38  //Set VDDA voltage
39
40      DrvPMU_VDDA_Voltage(E_VDDA2_4);         //VDDA=2.4
41      DrvPMU_VDDA_LDO_Ctrl(E_LDO);            //LDO ON
42      DrvPMU_BandgapEnable();
43      DrvPMU_REFO_Enable();
44
45      DrvPMU_AnalogGround(Enable);            //ADC analog ground source set
46                                              //1 : Enable buffer
47                                              //and use internal source
48                                              //need to work with ADC
49
50  //Set ADC interrupt
51
52      DrvADC_EnableInt();
53      DrvADC_ClearIntFlag();
54      DrvADC_Enable();
55
56      SYS_EnableGIE(7);                       //Enable GIE
57
58      while(1);                               //Wait for Interrupt
59
60  }
61
62  void HW2_ISR(void)                          //ADC interrupt
63  {
64      DrvADC_ClearIntFlag();                  //Clear ADC interrupt flag
65      ADCData=DrvADC_GetConversionData();     //Get ADC data
66  }
67
```

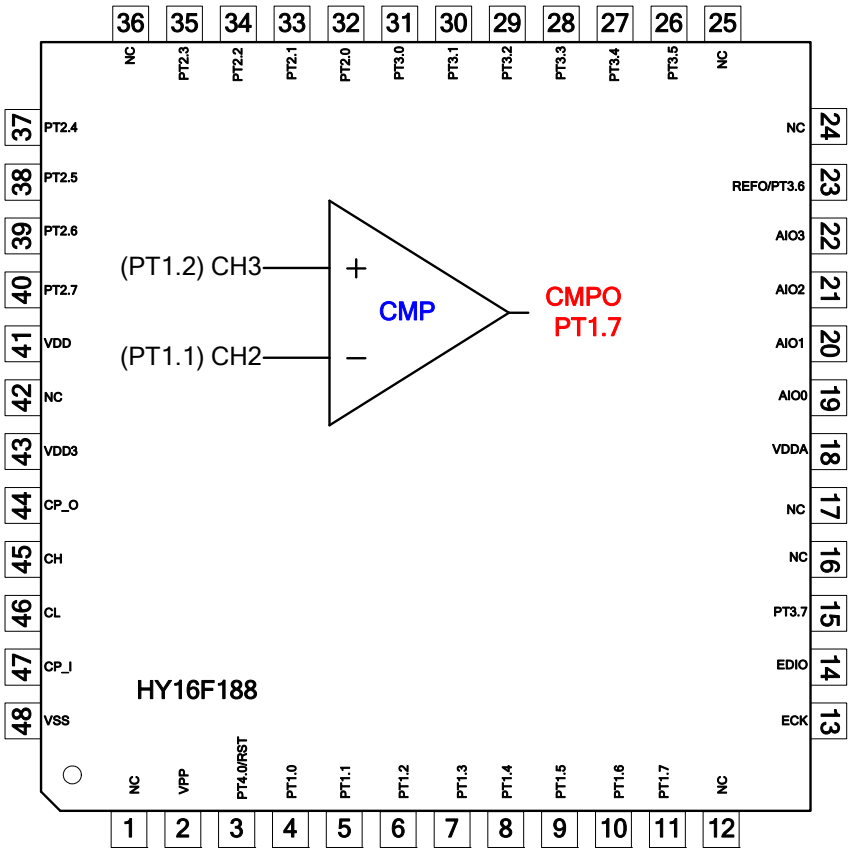
11. Analog IP (CMP)

11.1 Example Name

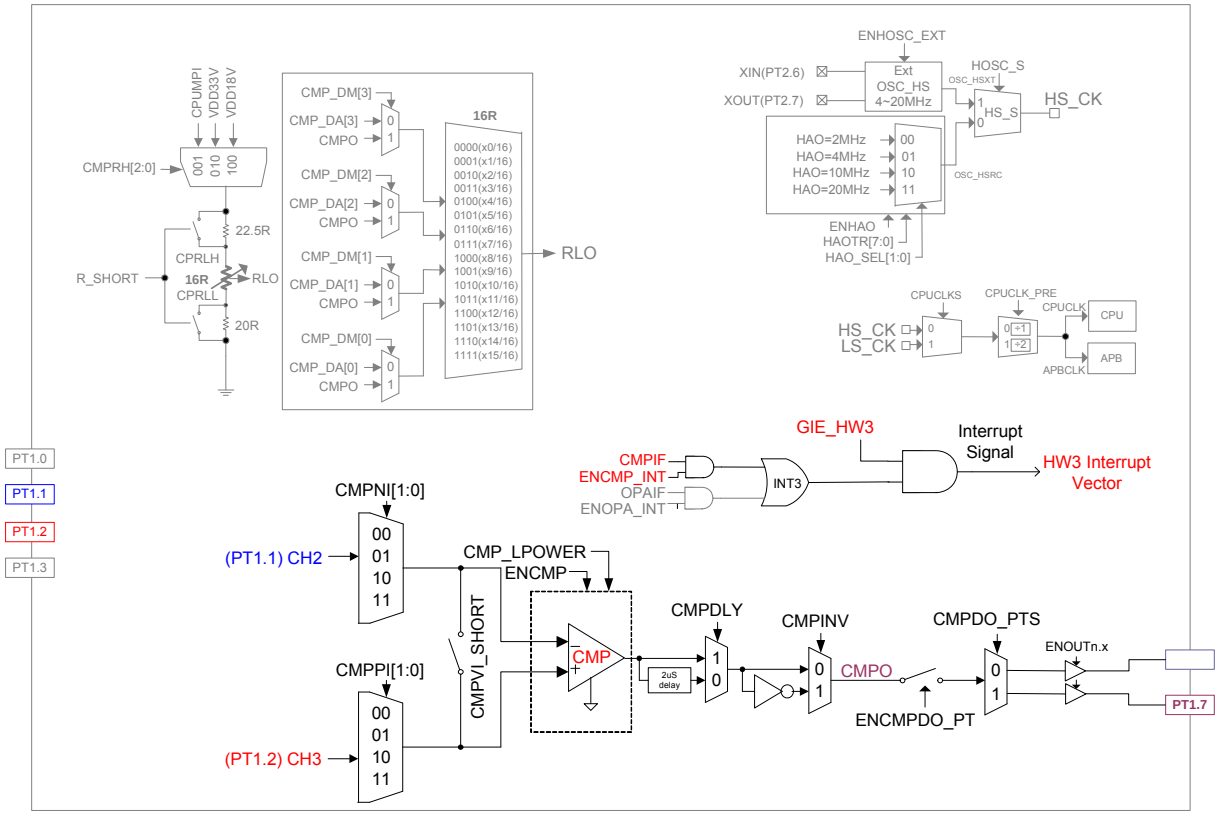
CMP usage

11.2 Example Description

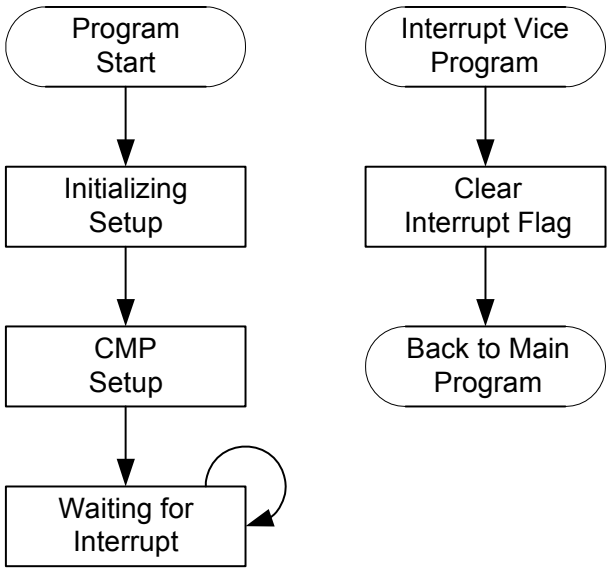
(1) Set up CMP register and IO.
(2) CH3(PT1.2) connects to CMP positive end.
(3) CH2(PT1.1) connects to CMPnegative end.
(4) If CH3 positive voltage larger than CH2 negative voltage, then CMPO digital output High.
(5) If CH3 positive voltage smaller than CH2 negative voltage, then CMPO digital output Low.



11.3 System Setup



11.4 Software Flow



11.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  int main(void)
04  {
05      DrvGPIO_Open(E_PT1,0X80,E_IO_OUTPUT);   //PT1.7 set Output
06      DrvCMP_PInput(2);                       //CMP positive input CH3
07      DrvCMP_NInput(1);                       //CMP negative input CH2
08
09      DrvCMP_Enable();                        //CMP enable
10      DrvCMP_OutputPinEnable(0);              //Enable CMP digital output to port
11                                              //0:PT1.7
12
13
14      DrvCMP_ClearIntFlag();                  //Clear CMP interrupt flag
15      DrvCMP_EnableInt();                     //Enable CMP Interrupt
16
17
18
19      SYS_EnableGIE(7);                       //Enable GIE (Global Interrupt Enable)
20      while(1);                               //while loop
21  }
22
23  void HW3_ISR(void)
24  {
25      DrvCMP_ClearIntFlag();                  //Clear CMP interrupt flag
26  }
27
```

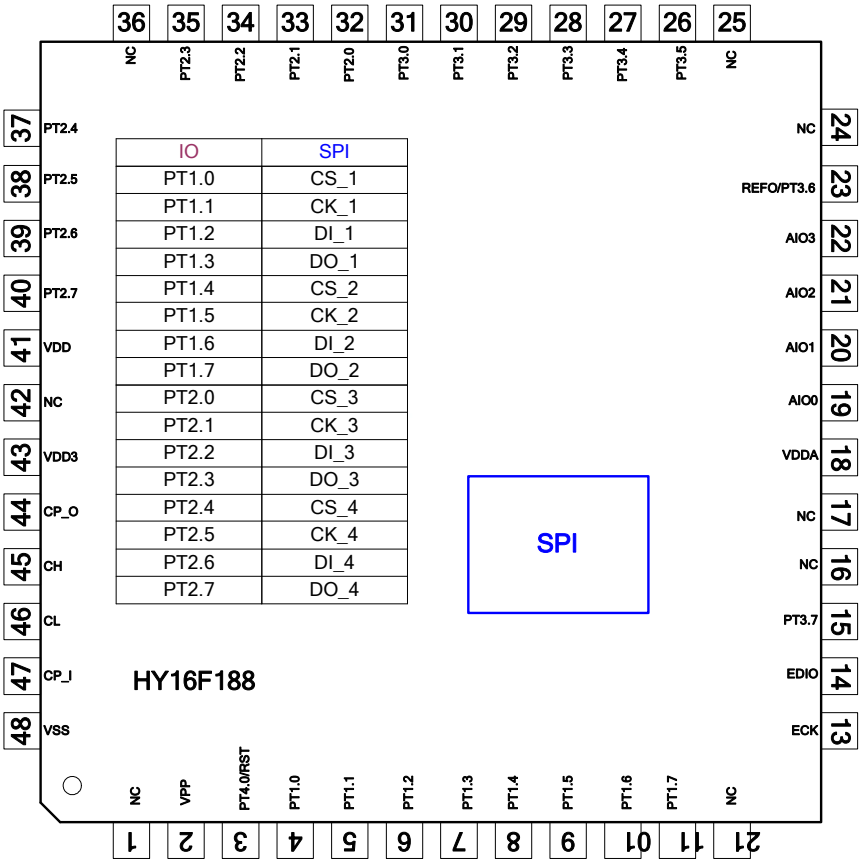
12. Communication IP (SPI)

12.1 Example Name

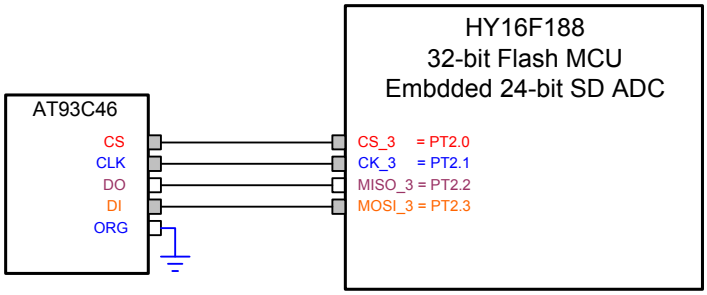
Communication SPI interrupt setup.

12.2 Example Description

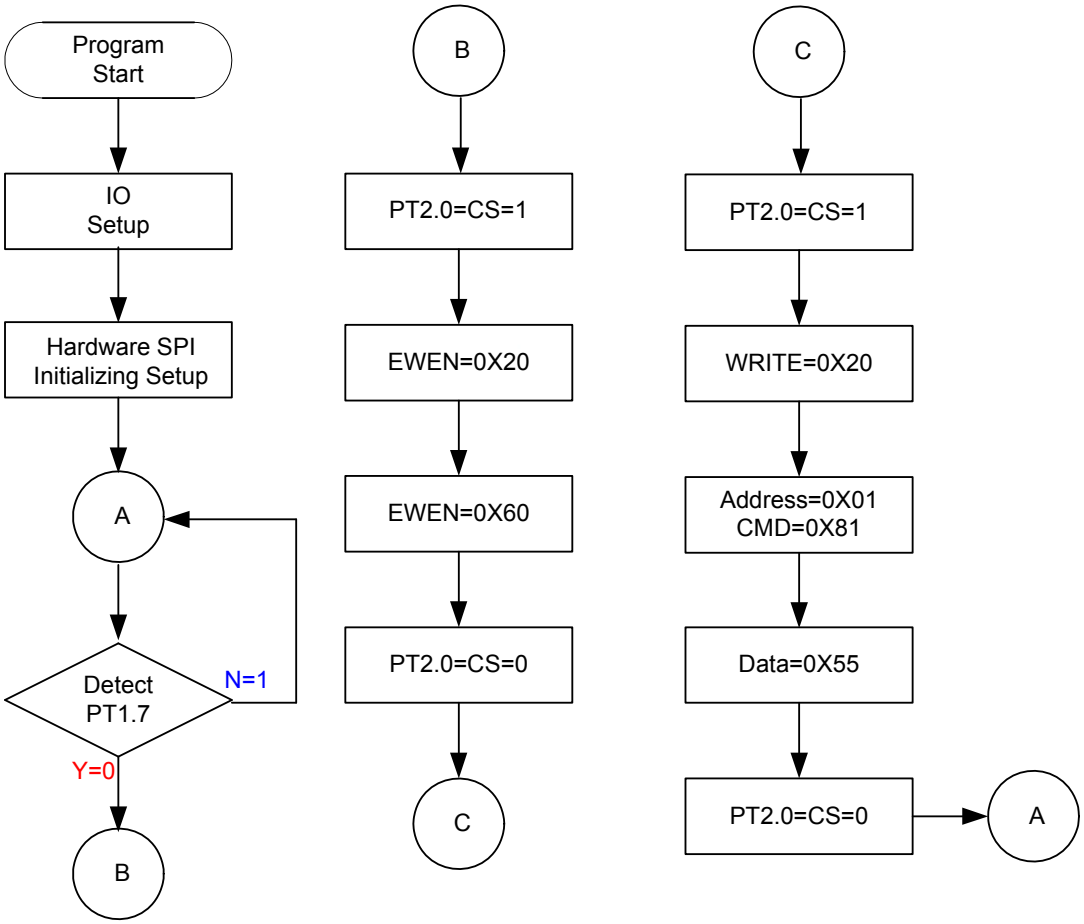
(1) SPI pin and SPI register setup
(2) Test hardware SPI to AT93C46 write-in function



12.3 System Setup



12.4 Software Flow



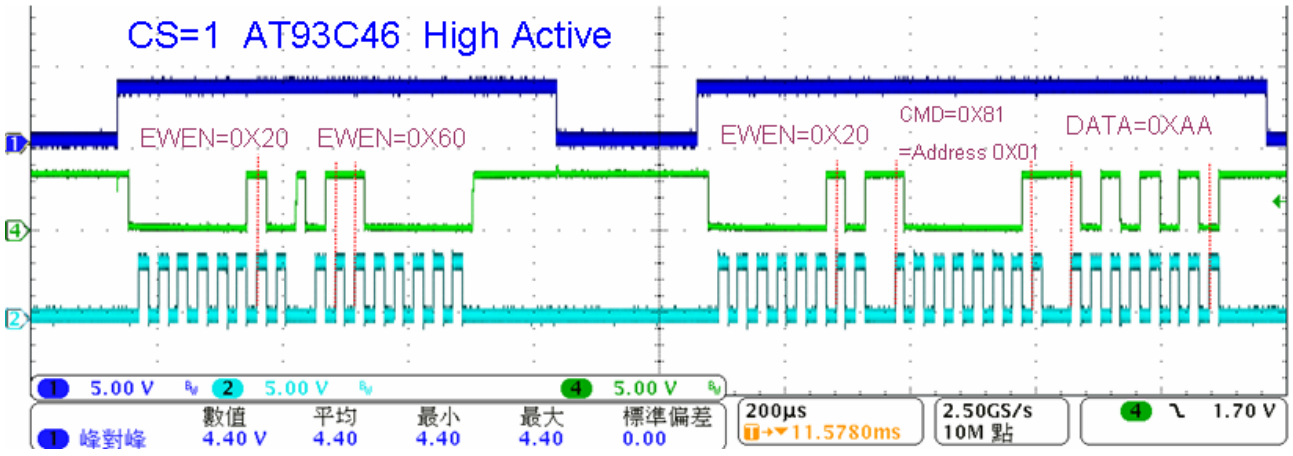
12.5 Program Description

(Main Program)

```
00
01  #include "HY16F18X.h"
02
03
04  volatile unsigned int i,a;
05
06  void InitalSPI32(void);
07  void Delay(unsigned int num);
08
09  int main(void)
11  {
12     SPI32_INI();
13
14     while(1)
15     {
16          i=DrvGPIO_GetBit(E_PT1,7);
17          while(i==0)
18          {
19                                          //Write enable
20              DrvGPIO_SetBit(E_PT2,0);    //PT2.0 CS=1
21              DrvSPI32_Write(0X02000000); //AT93C56=0X09
22              Delay(0X20);
23              DrvSPI32_Write(0X60000000); //AT93C56=0X80
24              Delay(0X50);
25              DrvGPIO_ClrBit(E_PT2,0);    //PT2.0 CS=0
26
27              Delay(0X20);
28
29
30                                          //Write CMD
31              DrvGPIO_SetBit(E_PT2,0);    //PT2.0 CS=1
32              DrvSPI32_Write(0X02000000); //AT93C56=0X0A
33              Delay(0X20);
34              DrvSPI32_Write(0X81000000); //AT93C56 Address 0X01
35              Delay(0X20);
36              DrvSPI32_Write(0X55000000); //AT93C56 Data 0X55
37              Delay(0X50);
38              DrvGPIO_ClrBit(E_PT2,0);    //PT2.0 CS=0
39
40              Delay(0X20);
41              i=DrvGPIO_GetBit(E_PT1,7);
42          }
43          Delay(0X200);
44      }
45      return 0;
46  }
47
48
```

(Vice Program)

```
00
01  void HW0_ISR(void)
02  {
03     int_00=0XFF00FF00;
04  }
05
06  void InitalSPI32(void)
07  {
08     //Set SPI input pin
09     DrvGPIO_Open(E_PT2,0X0B,E_IO_OUTPUT);          //CS=PT2.0 output
11     DrvGPIO_Open(E_PT2,0X04,E_IO_INPUT);
12     DrvGPIO_SetPortBits(E_PT2,0X01);
13
14     pio_44=0XFF50;
15     spi_00=0XFF00FF03;
16     spi_04=0XFF010007;
17     clk_0c=0XFF00FF0C;
18  }
19
20
21  void Delay(unsigned int num)                   //Software Delay Subroutines
22  {
23     volatile unsigned int d;
24     for(d=0;d<=num;d++);asm("NOP");
25  }
26
27
```



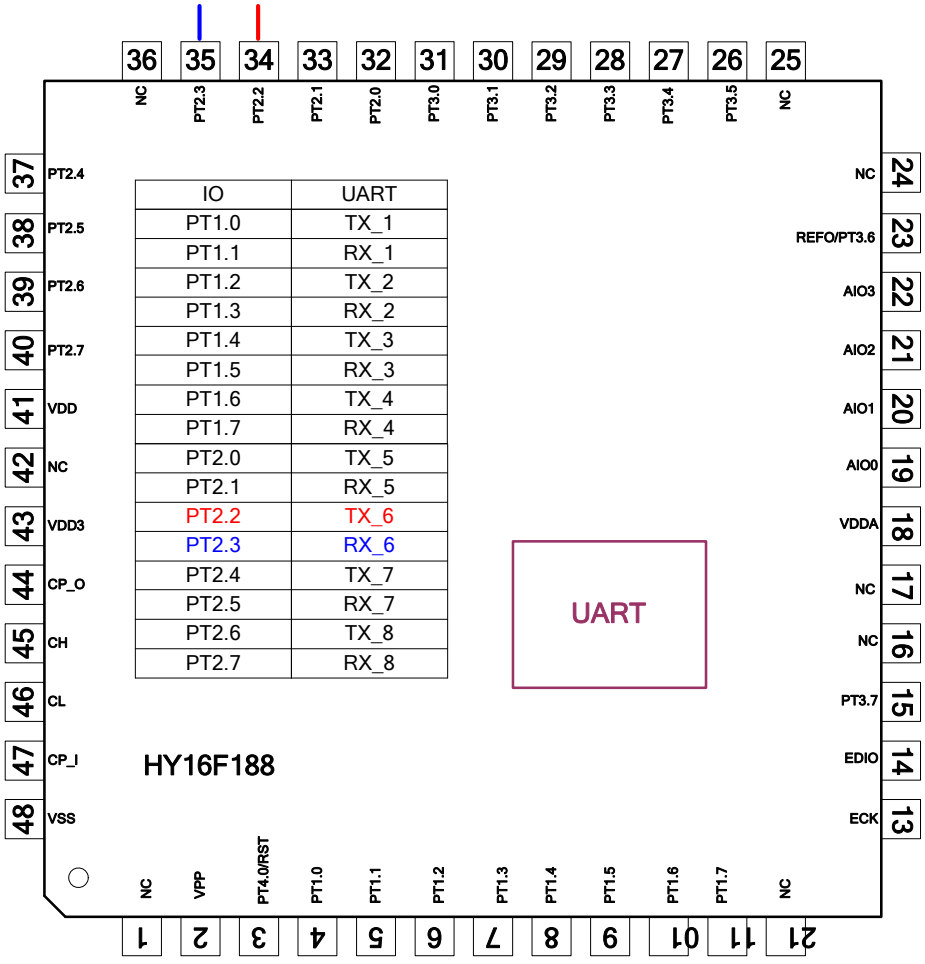
13. Communication IP (UART)

13.1 Example Name

Communication UART interrupt setup

13.2 Example Description

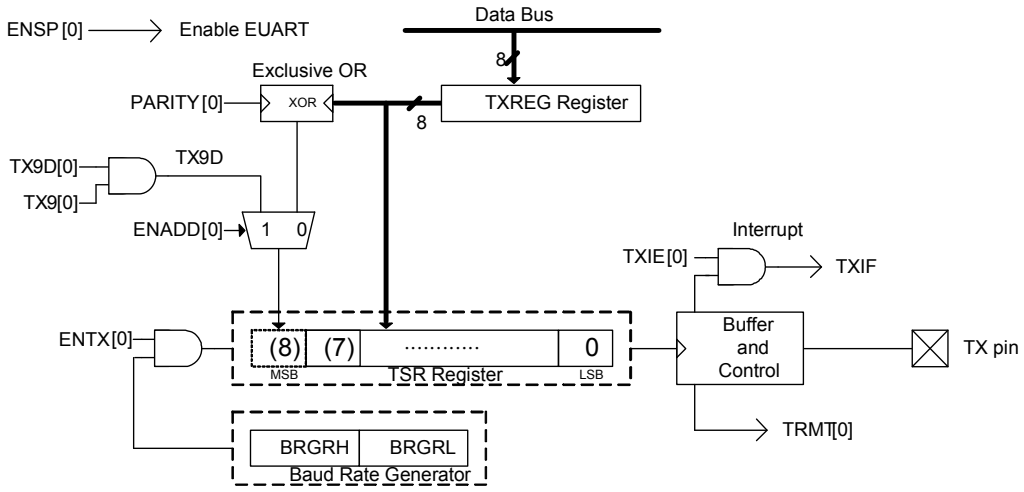
(1) Connecting TX and RX PIN to RS232 related circuit



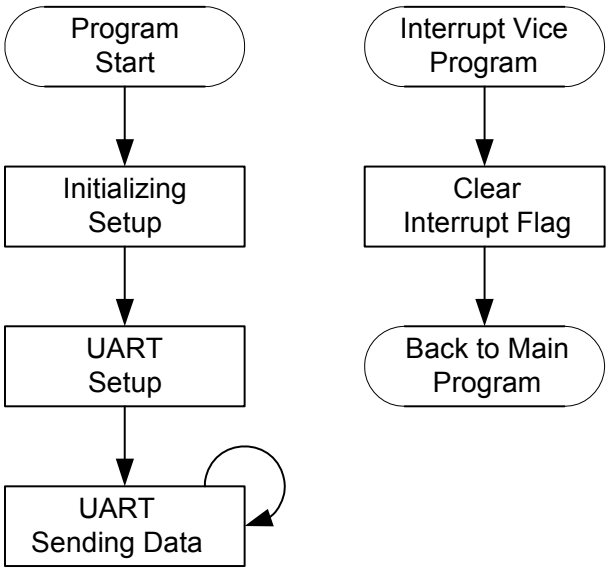
IO	UART
PT1.0	TX_1
PT1.1	RX_1
PT1.2	TX_2
PT1.3	RX_2
PT1.4	TX_3
PT1.5	RX_3
PT1.6	TX_4
PT1.7	RX_4
PT2.0	TX_5
PT2.1	RX_5
PT2.2	TX_6
PT2.3	RX_6
PT2.4	TX_7
PT2.5	RX_7
PT2.6	TX_8
PT2.7	RX_8

13.3 System Setup

EUART TRANSMIT BLOCK DIAGRAM



13.4 Software Flow



13.5 Program Description

```
00
01 #include "HY16F18X.h"
02 #include "DrvREG32.h"
03
04 void Delay (unsigned int num);
05 void All_in_One_Initial(void);
06
07 unsigned int temp,Buffer_Start,PT_1D7,i;                    //
08
09
10 int main(void)
11 {
12     All_in_One_Initial();                                    Initial UART and others
13     PT_1D7=DrvGPIO_GetBit(E_PT1,7);                          //
14     SYS_EnableGIE(7);
15     Buffer_Start=0x00;
16
17     while(1)
18     {
19         i=DrvGPIO_GetBit(E_PT1,7);                           //read PT1.7 pin
20         if(i==0)                                             //high or low
21         {                                                    //If PT1.7=0
22             int_00=0x08080c00;                               //UART INT set 0X08080C00
23             DrvUART_Write(Buffer_Start++);
24             Delay(0x8000);
25         }
26         Delay(0x8000);
27     }
28
29     return 0;
30 }
31
32 void HW0_ISR(void)                                          //UART Interrupt
33 {
34     int_00=0x0e000c00;                                       //Clear UART interrupt flag
35     temp=DrvUART_Read();
36     int_00=0x08000c00;                                       //Clear UART interrupt flag
37 }
39
```

```
40
41 void Delay(unsigned int num)
42 {
43     volatile unsigned int d;
44     for(d=0;d<=num;d++);asm("NOP");
45 }
46
47 void All_in_One_Initial(void)
48 {
49    DrvGPIO_Open(E_PT1,0x80,E_IO_INPUT);                                  //set PT1_7 INPUT
50    DrvGPIO_Open(E_PT1,0x80,E_IO_PullHigh);                               //enable PT1_7
51                                                                          //pull high R
52    DrvGPIO_Open(E_PT2,0x04,E_IO_OUTPUT);                                 //PT2.2 output TX.5
53    DrvGPIO_Open(E_PT2,0x08,E_IO_INPUT);                                  //PT2.3 input RX.5
54
55    DrvCLOCK_EnableHighOSC(E_EXTERNAL);                                   //select HSXT 4MHz
56    DrvUART_Open(4,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,5);
57
58    DrvUART_DisableAutoBaudrate();
59    clk_08=0xff10ff00;
60
61    asm volatile("sethi    $r0, 0xc0000");                                //N801 GIE setting1
62    asm volatile("ori      $r0, $r0, 0x003f");
63    asm volatile("mtsr     $r0, $INT_MASK");
64    asm volatile("movi     $r0, 0x70009");                                //N801 GIE setting2
65    asm volatile("mtsr     $r0, $PSW");
66 }
67
```

14. Communication IP (I^2C)

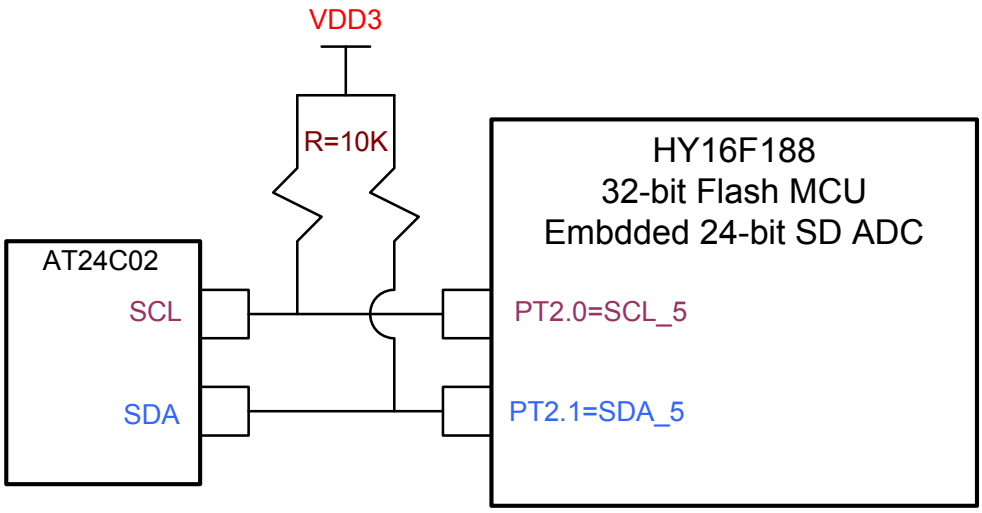
14.1 Example Name

Test hardware, I^2C

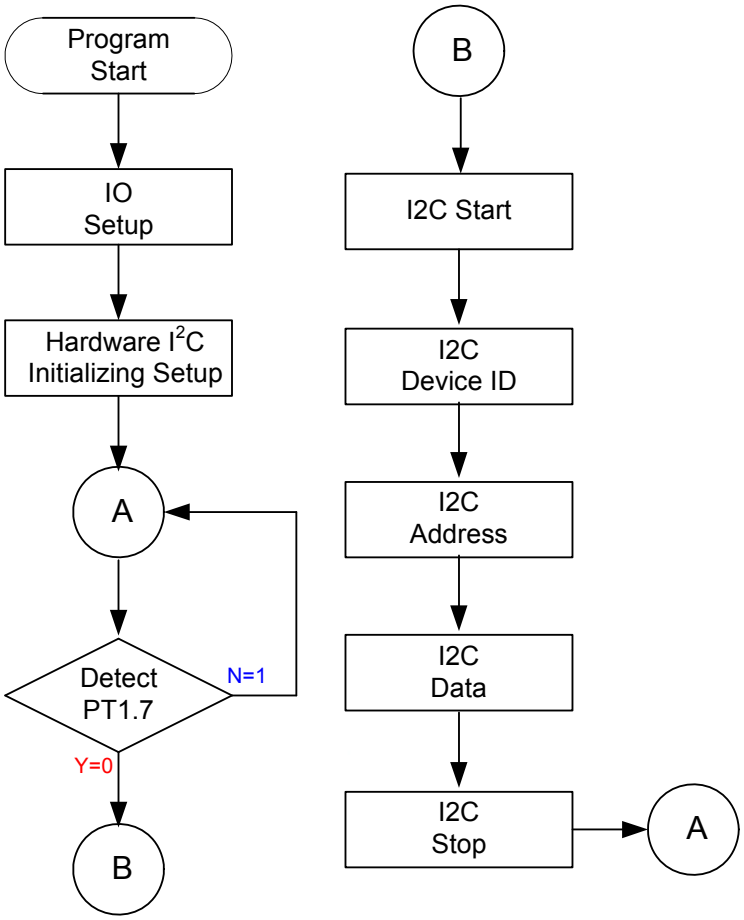
14.2 Example Description

(1) Set up I^2C register through I^2C pin.
(2) Test hardware I^2C write function to AT24C02

14.3 System Setup



14.4 Software Flow



14.5 Program Description

(Main Program)

```
00
01  #include "HY16F18X.h"
02
03
04  volatile unsigned int i,a;
05
06
07  int main(void)
08  {
09     I2C_INI();
11
12     while(1)
13     {
14          i=(pio_08 & 0x80)>>7;a=i;
15          while(a==0)
16          {
17               I2C_Start();                    //HY16F188 Hardware I2C Start
18
19               DrvI2C_WriteData(0XA0);         //AT24C02 Device ID
20               I2C_NACK();
21
22               DrvI2C_WriteData(0X00);         //AT24C02 Address=0X00
23               I2C_NACK();
24
25               DrvI2C_WriteData(0X55);         //Data1=0X55 @ Address=0X00
26               I2C_NACK();
27
28               DrvI2C_WriteData(0XAA);         //Data2=0XAA @ Address=0X01
29               I2C_NACK();
30
31               I2C_Stop();                     //HY16F188 Hardware I2C Stop
32
33               i=(pio_08 & 0x80)>>7;a=i;
34          }
35          Delay(0x50);
36      }
37      return 0;
38  }
```

(Vice Program)

```
00
01 void I2C_INI(void)                          //Hardware I2C Initial
02 {
03    pio_00=0XFF800000;                       //PT1.7 input
04    pio_04=0XFF800000;
05    pio_10=0XFF00FF03;
06    pio_14=0XFF00FF03;
07
08    pio_44=0X0909FF00;
09    i2c_00=0XFF00FF00;                       //0X41000//I2C OFF
11    i2c_14=0X0000FFFF;                       //0X44014
12 }
13
14
15 void I2C_Start(void)                         //Hardware I2C Start
16 {
17    DrvI2C_Ctrl(1,0,0,0);                    //SPIA(1000)
18    I2C_NOP();
19 }
20
21 void I2C_Stop(void)                          //Hardware I2C Stop
22 {
23    DrvI2C_Ctrl(0,1,0,0);                    //SPIA(0100)
24    I2C_NOP();
25 }
26
27 void I2C_Write(unsigned int I2C_Data)        //Hardware I2C Write
28 {
29    DrvI2C_WriteData(I2C_Data);
30    i2c_04=0XFF01;                           //0X41004
31    I2C_NOP();
32 }
33
34 void I2C_NACK(void)                          //Hardware I2C NACK
35 {
36    DrvI2C_ClearIRQ();
37    DrvI2C_ClearEIRQ();
38    while(!(i2c_04 & 0X00000002));
39 }
40
41 void I2C_NOP(void)                           //Software I2C NOP
42 {
43    volatile unsigned int n;
44    for(n=0;n<=0x10;n++);asm("NOP");//0x10
45 }
46
```

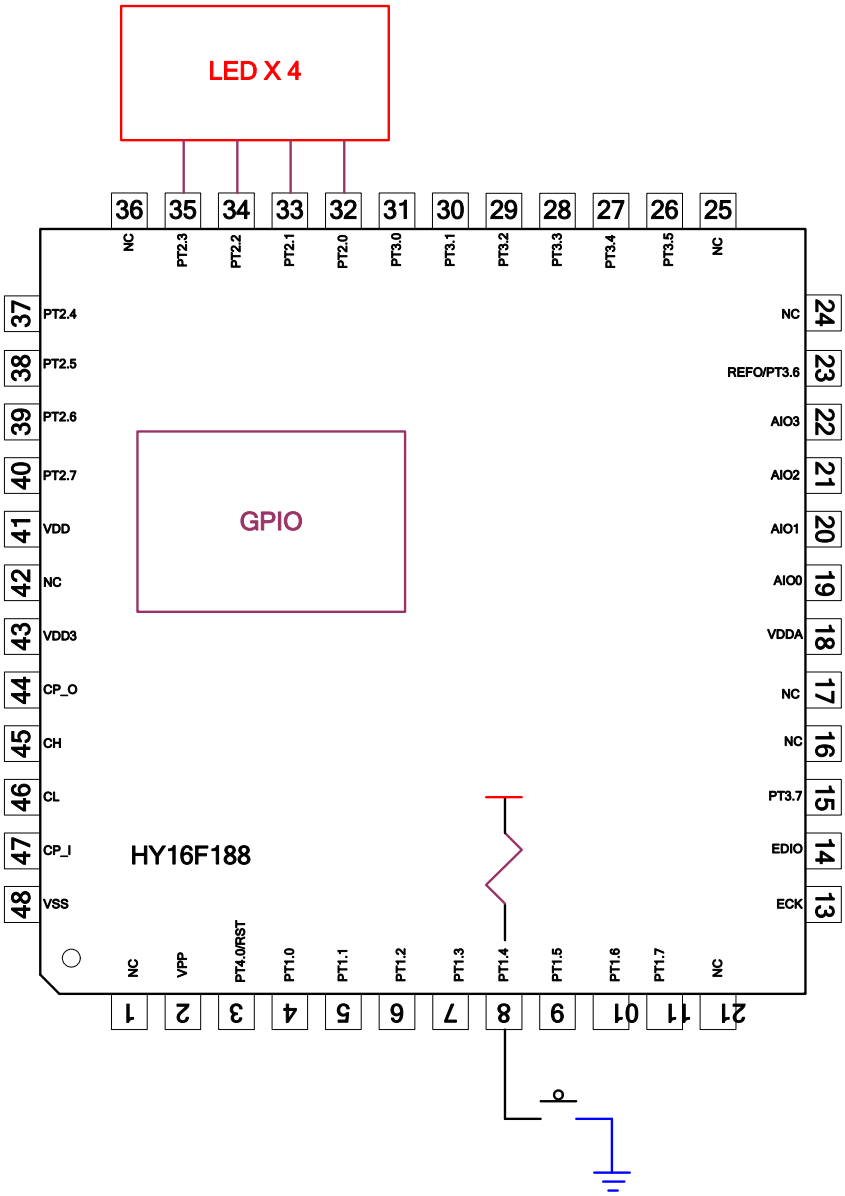
15. Peripheral IP(GPIO)

15.1 Example Name

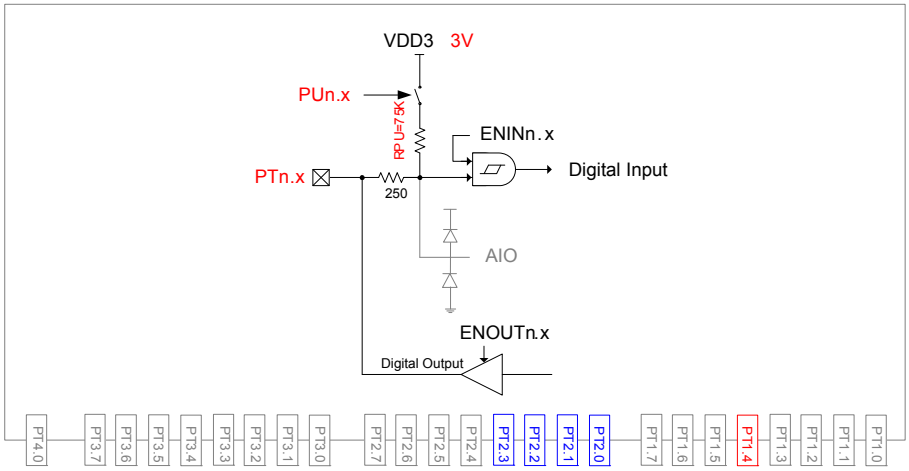
GPIO configurations

15.2 Example Description

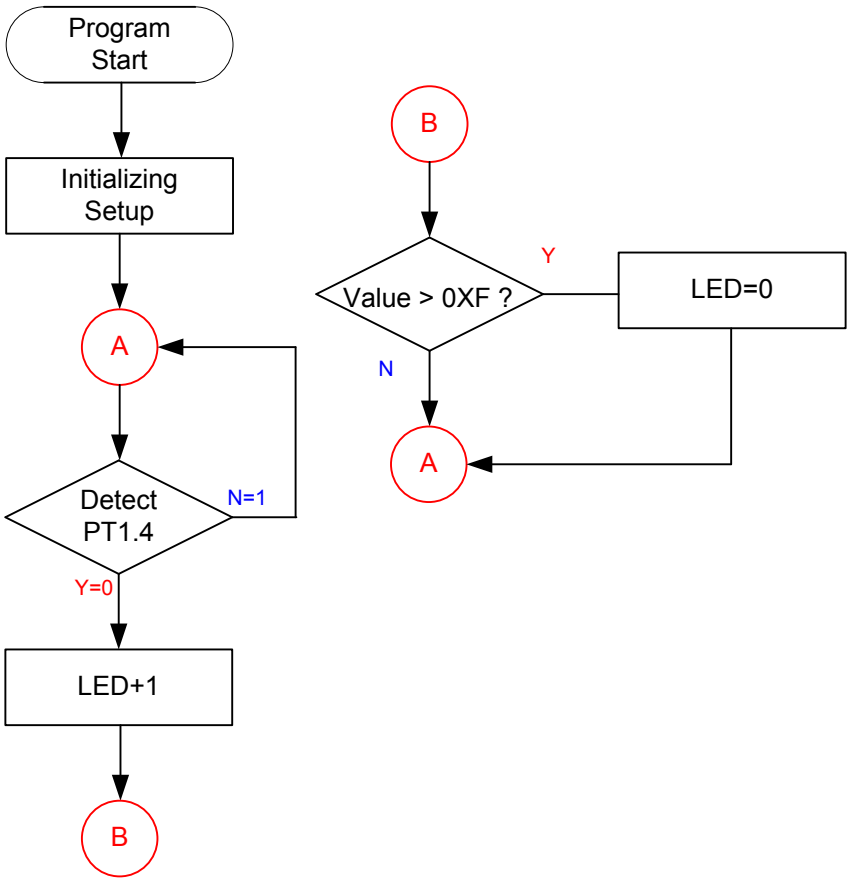
(1) Everytime when PT1.4 is pressed, LED display adds 1.
(2) If LED increased to 0XF, then cleared 0. LED display adds 1 when PT1.4 pressed.



15.3 System Setup



15.4 Software Flow



15.5 Program Description

```
00
01  #include "HY16F188.h"
02
03  void Delay (unsigned int num);
04
05  int main(void)
06  {
07      unsigned int i=0,j=0;
08      DrvGPIO_Open(E_PT1,0X10,E_IO_INPUT);          //Set PT1_4 INPUT
09      DrvGPIO_Open(E_PT1,0X10,E_IO_PullHigh);       //Enable PT1 4 pull hi R
11      DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);         //Set PT2_0~3 OUTPUT
12
13      while(1)
14      {
15          i=DrvGPIO_GetBit(E_PT1,4);                //Read PT1.4 high or low
16          if(i==0)                                  //IF PT1.4 is low
17          {
18              DrvGPIO_SetPortBits(E_PT2, j++);      //J++
19              if(j>0X0F)j=0X00;                     //IF J>0XF J=0
20          }
21
22      Delay(0X8000);
23  }
24
25  void Delay(unsigned int num)
26  {
27      int a;for(a=0;a<=num;a++);                    //Delay loop
28  }
29
```

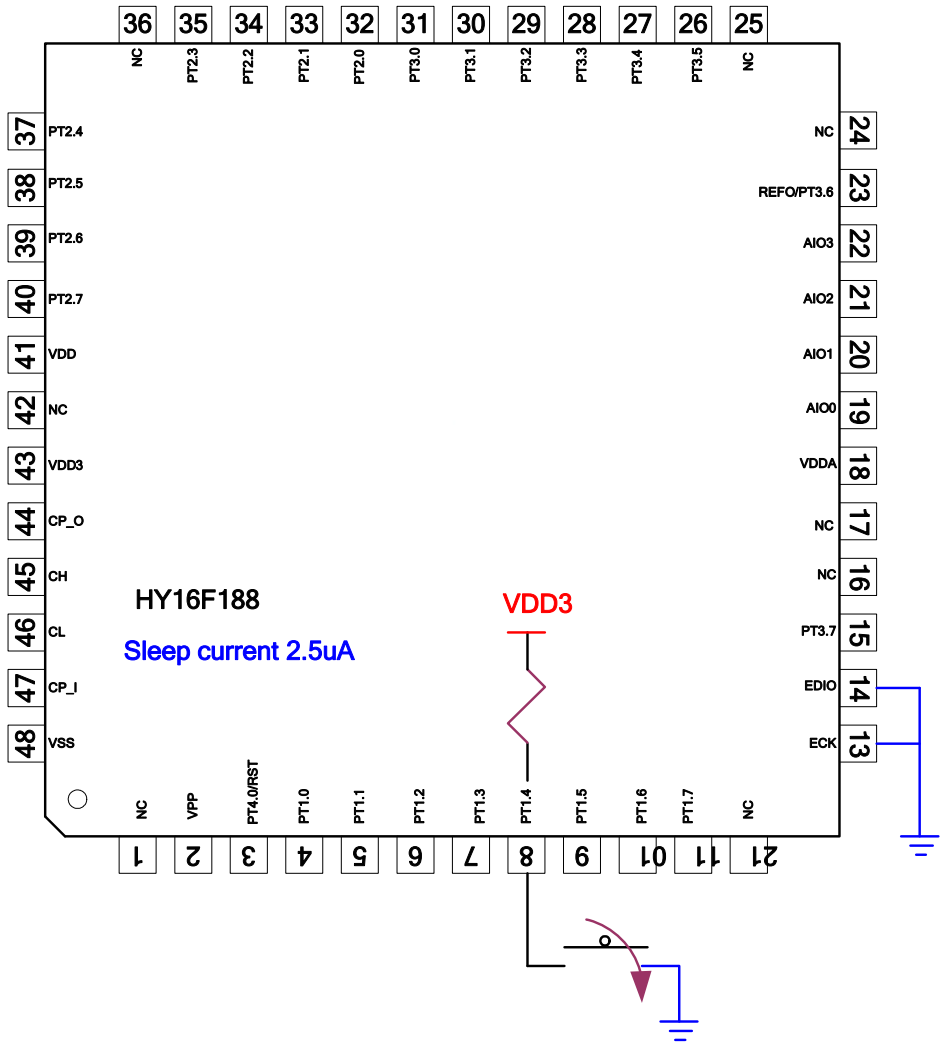
16. Peripheral IP (Power)

16.1 Example Name

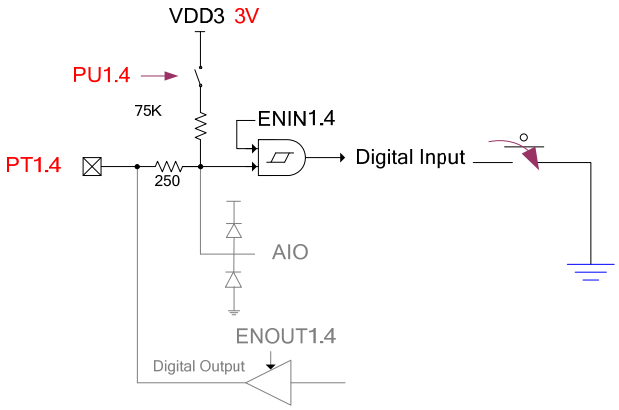
Test Sleep and Idle current

16.2 Example Description

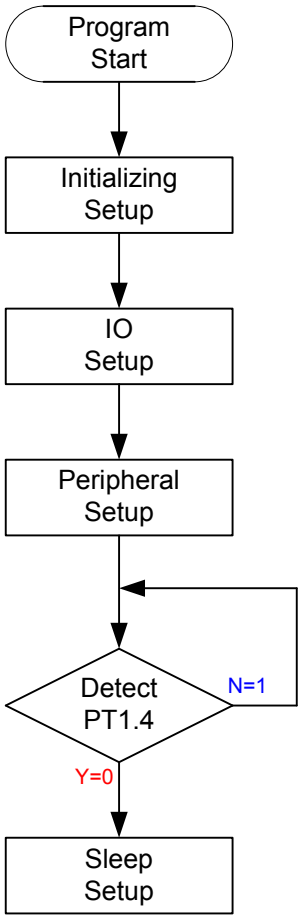
(1) Do not float every pin. Ex. If grounding 2PIN (the 13rd and 14th pin) of EDM, Sleep current can be lowered than 2.5 (uA).
(2) IC will enter into Sleep mode when PT1.4 is pressed.



16.3 System Setup



16.4 Software Flow



After this program entered into Sleep, current measured can be lowered than 2.5 (uA) min..
To awaken the IC, only Power ON Reset can be used.

16.5 Program Description

```
00
01  #include "HY16F18X.h"
02
03  void Delay (unsigned int num);
04
05  int main(void)
06  {
07
08      DrvGPIO_ClearIntFlag(E_PT2,0 XFF);          //clear PT2.0~7 interrupt flag
09
11      DrvGPIO_Open(E_PT1,0X3F,E_IO_OUTPUT);       //SET PT1.0~6 AS OUTPUT PT1.6~7 INPUT
12      DrvGPIO_Open(E_PT3,0XFF,E_IO_OUTPUT);       //SET PT3 AS OUTPUT
13
14      DrvGPIO_Open(E_PT1,0XC0,E_IO_INPUT);        //PT1_6~7 set input
15      DrvGPIO_Open(E_PT1,0XC0,E_IO_PullHigh );    //PT1_6~7 pull internal high R enable
16
17      DrvGPIO_Open(E_PT2,0XFF,E_IO_INPUT);        //PT2_0~7 set input
18      DrvGPIO_Open(E_PT2,0XFF,E_IO_PullHigh );    //PT2_0~7 pull internal high R enable
19
20      DrvPMU_LDO_LowPower(1);                     //SET low power mode
21      i=DrvGPIO_GetBit(E_PT1,7);                  //read PT1.7 pin high or low
22
23      DrvCLOCK_EnableHighOSC(E_INTERNAL);         //select HSRC 2MHz
24      DrvGPIO_ClkGenerator(E_HS_CK,1);            //GPIO CLK enable
25                                                  // trigger method is negative edge
26      DrvGPIO_IntTrigger(E_PT2,0XFFE_N_Edge);     //PT2.0~7 interrupt
27
28      DrvGPIO_Open(E_PT2,0XFF,E_IO_IntEnable);    //PT2.0~7 interrupt enable
29
30      SYS_EnableGIE(7);                           //Enable GIE (Global Interrupt Enable)
31      asm("sethi $r0,0xc0000");
32      asm("ori      $r0,$r0,0x003f");
33      asm("mtsr    $r0,$ir14");
34
35
```

```
36
37    while(1)
38    {
39        i=DrvGPIO_GetBit(E_PT1,7);          //read PT1.7 pin high or low
40        if(i==0)                            //if PT1.7 low
41        {
42            clk_08=0XFF80FF03;              //[CPU/2][RTC Clock On]
43            clk_00=0XFF00FF04;              //[Internal OSC OFF][32768]
44            asm("NOP");
45            sys_04=0X1000;                  //sleep mode
46            //sys_04=0X1010;                //idle mode
47            asm("standby 1");
48            asm("NOP");
49        }
50        Delay(0X8000);
51    }
52 }
53 void HW5_ISR(void)
54 {
55    DrvGPIO_ClearIntFlag(E_PT2,0XFF);       //Clear PT2 interrupt flag
56    asm("sethi    $r0,0xc0000");
57    asm("ori      $r0,$r0,0x003f");
58    asm("mtsr     $r0,$ir14");
59 }
60
61 void Delay(unsigned int num)
62 {
63    int a;for(a=0;a<=num;a++);
64 }
65
```

SYS Base Address + 0X04 (0X40104)																	
Symbol	SYS0 (SYS Control Register 0)																
Bit	[31]	[30]	[29]	[28]	[27]	[26]	[25]	[24]	[23]	[22]	[21]	[20]	[19]	[18]	[17]	[16]	
Name	RSV																
RW	R-0																
Bit	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]	
Name	MASK								-	-	F_{CRst}	IDLE	$F_{SLP/IDLE}$	F_{WDog}	F_{RST}	F_{BOR}	
RW	R0W-0								-	-	RW0-0					RW0-1	

Bit	Name	Descriptions	
Bit[4]	IDLE	IDLE Mode Control Register	
		0	Sleep Mode
		1	IDLE Mode

(1) Sleep Mode

```
asm("standby 1");
sys_04=0X1000;
```

(2) Idle Mode

```
asm("standby 1");
sys_04=0X1010;
```

17. Digital IP(RTC)

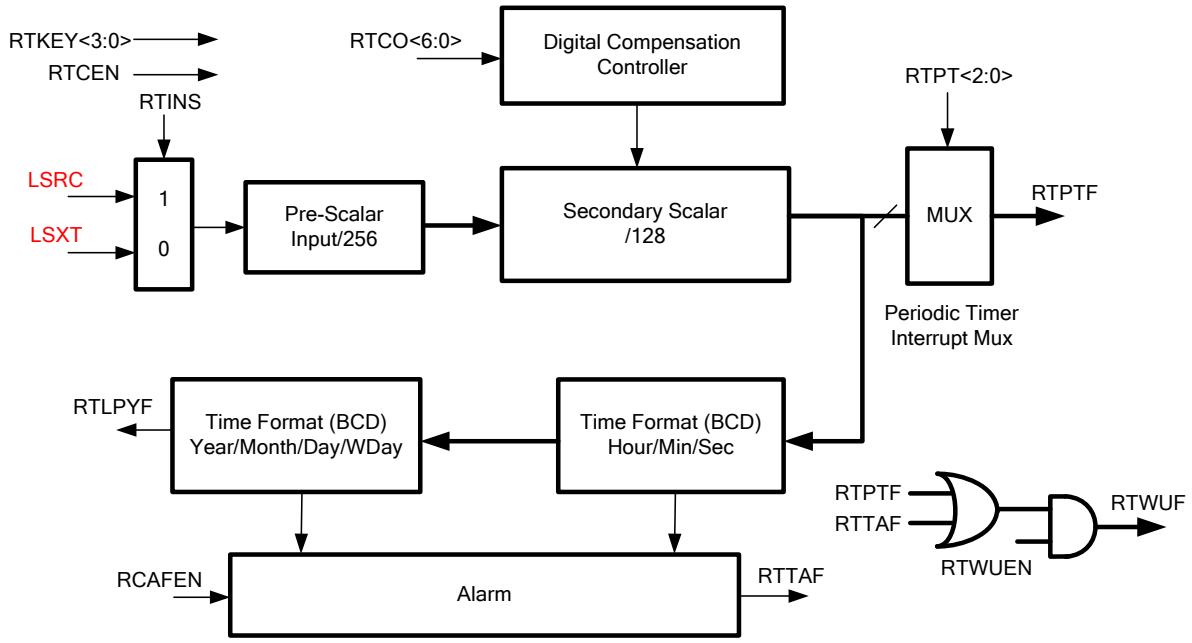
17.1 Example Name

Test hardware RTC

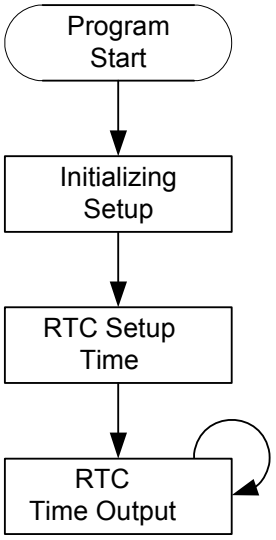
17.2 Example Description

Test hardware RTC calendar function

17.3 System Setup



17.4 Software Flow



17.5 Program Description

(Main Program)

```
00
01  #include "HY16F18X.h"
02  unsigned int sec,min,hour,week,day,month,year;
03
04  void Delay(unsigned int num);
05  void RTC_Initial(void);
06
07  int main(void)
08  {
09     RTC_Initial();                                  //RTC initializing(time setup included)
11
12     while(1)
13     {
14        asm("NOP");
15        S_DRVRTC_TIME_DATA_T sCurTime;               //RTC read time setup
16        DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);  //RTC read time data
17
18        sec=sCurTime.u32cSecond;                     //read_second
19        min=sCurTime.u32cMinute;                     //read_minute
20        hour=sCurTime.u32cHour;                      //read_hour
21        week=sCurTime.u32cDayOfWeek;                 //read_week
22        day=sCurTime.u32cDay;                        //read_date
23        month=sCurTime.u32cMonth;                    //read_month
24        year=sCurTime.u32Year;                       //read_year
25        asm("NOP");
26     }
27     return 0;
28  }
29
```

(Vice Program)

```
00
01  void RTC_Initial(void)
02  {
03     //RTC CLK;
04     clk_08=0x8080ff00;
05     //RTC KEY;
06     rtc_00=0xff60ff00;
07     asm("NOP");
08
09     DrvRTC_WriteEnable();
11     //DrvRTC_ClockSource(E_INTERNAL_CLOCK);//35KHz
12     DrvRTC_ClockSource(E_EXTERNAL_CLOCK);//32768Hz
13
14     DrvRTC_PeriodicTimeEnable(0);//set 1/128
15     DrvRTC_Enable();
16     DrvRTC_HourFormat(0);
17
18     asm("NOP");
19     S_DRVRTC_TIME_DATA_T sCurTime;//setting start
20     DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
21     sCurTime.u8cClockDisplay=1;
22     sCurTime.u8cAmPm=0;
23     sCurTime.u32cSecond=19;                  //Setup_second
24     sCurTime.u32cMinute=50;                  //Setup _minute
25     sCurTime.u32cHour=10;                    //Setup _hour
26     sCurTime.u32cDayOfWeek=5;                //Setup_week
27     sCurTime.u32cDay=9;                      //Setup_date
28     sCurTime.u32cMonth=8;                    //Setup_month
29     sCurTime.u32Year=2013;                   //Setup_year
30     sCurTime.u8IsEnableWakeUp=0;
31     DrvRTC_Write(DRVRTC_CURRENT_TIME,&sCurTime);
32     asm("NOP");
33  }
34
```

18. Appendix Program

	Digital IP	Analog IP	Communication IP	Peripheral IP
01	TimerA	8-bit DAC	Hardware 32-bit SPI	GPIO
02	TimerB	R2R OPAMP	Hardware UART	Sleep Mode
03	TimerC	24-bit SD ADC	Hardware I2C	Idle Mode
04	WDT	Analog CMP	–	–
05	PWM	–	–	–
06	Hardware RTC	–	–	–


HY16F18X_20130910.rar

19. Revision History

Major amendment is stated thereinafter:

Version	Page	Revision Summary	Date
V03	ALL	First Edition	2013/12/1