HY16F18 Series
HYCON IP User's Manual

Table of Contents

Attention:

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

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

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

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

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

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

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

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

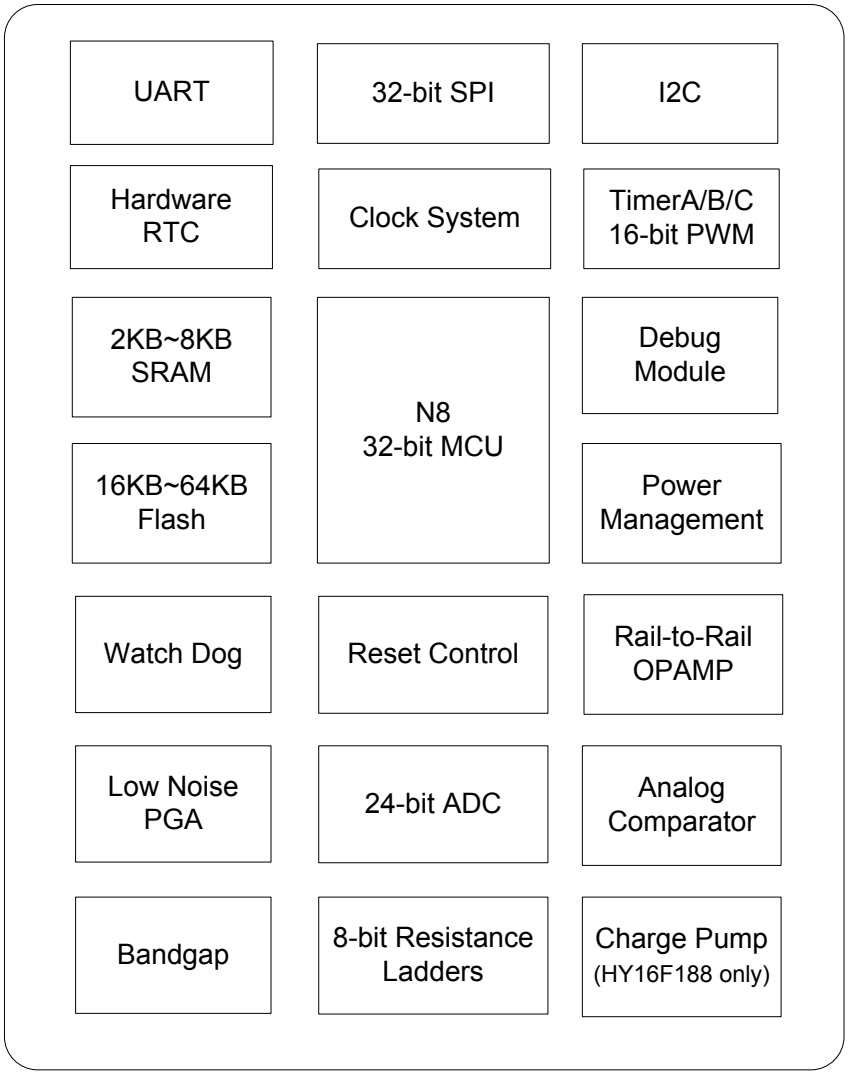
1. Document Description

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 HY16F18 Series, of which can be grouped into four categories:

(1) Digital IP : TimerA/TimerB/TimerC/WDT/PWM/Hardware RTC/GPIO
(2) Analog IP : 8 bit Resistance Ladder(DAC)/ADC/OPAMP/CMP
(3) Communication IP : Hardware 32-bit SPI/ Hardware UART/ Hardware I2C
(4) Peripheral IP : Power Management

2. IC Description

Basic description of each HY16F18X IP.



(01) Adopting Andes Technology 32-bit CPU core, N801 processor.
(02) Voltage operation range: 2.2~3.6V(No analog power enable condition), and temperature operation range: -40℃~85℃.
(03) Support external 16MHz crystal oscillator or internal 10MHz RC oscillator,
(04) Program memory: 64K-Byte Flash ROM
(05) Data memory: 8K-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
(09) Built-in hardware 8-bit Resistance Ladder (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 function
(13) Hardware serial communication 32-bit SPI/I2C/UART module
(14) Hardware RTC clock function module

※Interrupt control system

Interrupt Vector Address	N801	Interrupt Function
INT Base Address + 0x00 (COM)	HW0	void HW0_ISR(void)
INT Base Address + 0x04 (Timer ABC WDT HW RTC)	HW1	void HW1_ISR(void)
INT Base Address + 0x08 (ADC)	HW2	void HW2_ISR(void)
INT Base Address + 0x0C (CMP/OPA)	HW3	void HW3_ISR(void)
INT Base Address + 0x10 (PT1)	HW4	void HW4_ISR(void)
INT Base Address + 0x14 (PT2)	HW5	void HW5_ISR(void)

3. Digital IP (Timer)

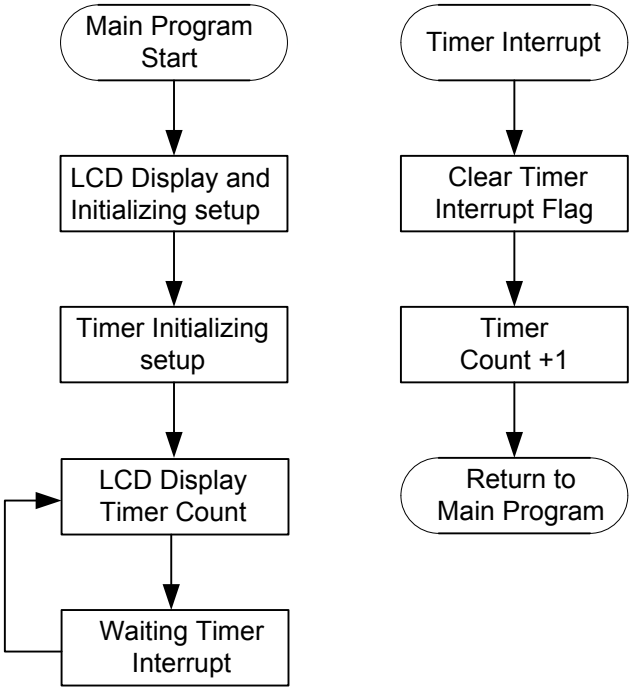
3.1. Example Name

HY16F188_Timer

3.2. Example Description

(1) Timer A/Timer B/Timer C tutorial.
(2) Using #define to select to compile TMATEST or TMBTEST or TMCTEST.
(3) In this example code, set Timer initializing and Timer overflow condition, enable System GIE and wait Timer interrupt. Timer overflow can decide Timer interrupt frequency.
(4) Everytime Timer interrupt occurs, in the Timer interrupt service routine, variable "Timer Count" to do +1. And then return to main program to show "Timer Count" on LCD.

3.3. Software Flowchart



3.4. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------------
 * Project Name : HY16F188_Timer
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *         GND    |
 *                |
 * ---------
 *****************************************************************************/
/*----------------------------------------------------------------------------*/
/* Includes                                                                   */
```

```
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvTimer.h"
#include "DrvI2C.h"
#include "DrvCLOCK.h"
#include "HY2613.h"
#include "my define.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char    _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define   I2CBufferSize   64
#define   TMATEST
//#define   TMBTEST
//#define   TMCTEST

/*---------------------------------------------------------------------------*/
```

```
/* Global CONSTANTS                                                    */
/*---------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;
unsigned int TimerA_count, TimerB_count, TimerC0_count, TimerC1_count;
/*---------------------------------------------------------------------*/
/* Function PROTOTYPES                                                 */
/*---------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);

/*---------------------------------------------------------------------*/
/* Main Function                                                       */
/*---------------------------------------------------------------------*/
int main(void)
{

   InitalI2C();
   SYS_EnableGIE(4,0x3F);                    // Enable GIE(Global Interrupt)
   DisplayInit();
   ClearLCDframe();
   Delay(10000);
   DisplayHYcon();
   Delay(10000);
   MCUSTATUSbits._byte = 0;

   TimerA_count=0;
   TimerB_count=0;
   TimerC0_count=0;
   TimerC1_count=0;

#if defined(TMATEST)
   DrvTMA_Open(15,0);                        //Timer A Overflow
```

```
                                        //15:taclk/65536/32;TMRDV=rup
                                        //00:HS_CK
    DrvTIMER_ClearIntFlag(E_TMA);       //Clear Timer A interrupt flag
    DrvTIMER_EnableInt(E_TMA);          //Timer A interrupt enable
#endif

#if defined(TMBTEST)
    DrvTMBC_Clk_Source(0,3);            //Timer B Prescaler 1
                                        //0: HS_CK,clock source.
                                        //3: clock divider.ur

    DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0xFFFF);   //Timer B overflow 0xffff
    DrvTIMER_ClearIntFlag(E_TMB);       //Clear Timer B interrupt flag
    DrvTIMER_EnableInt(E_TMB);          //Timer B interrupt enable

#elif defined(TMCTEST)
    DrvTMBC_Clk_Source(0,1);            //Timer B Prescaler 1
                                        //0: HS_CK,clock source.
                                        //1: clock divider.ur

    DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0xFFFF);   //Timer B overflow 0xffff

    DrvCapture1_Open(2,14,1);           //Timer C0 use as Capture 1
                                        //input source selection
                                        //2:LS_CK
                                        //14:S_CKource selectione 1lag)er
    DrvCapture2_Open(1,1);              //Timer C1 use as Capture 2
                                        //Input source selection
                                        //1:same as Caputre1
                                        //Falling-edge trigger

    DrvTIMER_ClearIntFlag(E_TMC0);      //Clear TMC0 interrupt flag
    DrvTIMER_ClearIntFlag(E_TMC1);      //Clear TMC1 interrupt flag
    DrvTIMER_EnableInt(E_TMC0);         //Timer C0 interrupt enable
    DrvTIMER_EnableInt(E_TMC1);         //Timer C1 interrupt enable
#endif

    SYS_EnableGIE(4,0x3F);              //Enable GIE(Global Interrupt)
    MCUSTATUSbits._byte = 0;
```

```
    while(1)                                        //Wait for Interrupt
    {

    if(MCUSTATUSbits.b_TMAdone==1)
    {
       LCD_DATA_DISPLAY(TimerA_count);
       MCUSTATUSbits.b_TMAdone=0;
    }

    if(MCUSTATUSbits.b_TMBdone==1)
    {
       LCD_DATA_DISPLAY(TimerB_count);
       MCUSTATUSbits.b_TMBdone=0;
    }

    if(MCUSTATUSbits.b_TMC0done==1)
    {
       LCD_DATA_DISPLAY(TimerC0_count);
       MCUSTATUSbits.b_TMC0done=0;
    }

    if(MCUSTATUSbits.b_TMC1done==1)
    {
       LCD_DATA_DISPLAY(TimerC1_count);
       MCUSTATUSbits.b_TMC1done=0;
    }

    }

    return 0;
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
```

```
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
  unsigned char I2C_Status,I2C_IntFlag;

  I2C_IntFlag=DrvI2C_ReadIntFlag();
  if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
  {
    I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
    switch(I2C_Status)
    {
      case 0x90:   //MACTFlag+RWFlag
          {          //START has been transmitted
            DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
            break;
          };
      case 0x84:   //MACTFlag+ACKFlag
          {          //Slave A + W has been transmitted. ACK has been received.
            DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;
          };
      case 0x80: //MACTFlag
          {         //Slave A + W has been transmitted. ACK has been received.
            DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;
          };
      case 0x30:
          {
            DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
            I2C_EndFlag=1;
            break;
          };
      case 0x8C:   //MACTFlag+DFFlag+ACKFlag
          {          //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
```

```
            DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
          }
          else
          {
            if(I2C_RW == I2C_WRITE)
            {
              DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
              I2C_EndFlag=1;
            }
            else if(I2C_RW == I2C_READ)
              DrvI2C_Ctrl(1,0,0,0);    //I2C as master sends START signal
            I2C_DataTxIndex=0;
          }
          break;
        };
    case 0x88:    //MACTFlag+DFFlag
        {         //DATA has been transmitted and NACK has been received
          DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
          I2C_DataTxIndex=0;
          I2C_EndFlag=1;
          break;
        };
    case 0xB0:
        {         //A repeated START has been transmitted.
          DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
          DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
          break;
        }
    case 0x94:    //MACTFlag+RWFlag+ACKFlag
        {         //Slave A + R has been transmitted. ACK has been received.
          if(I2C_DataRxLen>1)
          {
            DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
          }else{
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          }
          break;
        };
```

```
        case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
            {           //Data byte has been received. ACK has been transmitted.
              I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
              if((I2C_DataRxLen-1)>I2C_DataRxIndex)
              {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
              }
              else
              {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
              }
              break;
            };
        case 0x98:    //MACTFlag+RWFlag+DFFlag
            {           //Data byte has been received. NACK has been transmitted.
              I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
              DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
              I2C_EndFlag=1;
              break;
            };
        default:
            {
              DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
              I2C_EndFlag=1;
              break;
            };
    }
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
    DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
}
if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
{
    I2C_EndFlag=1;
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
    DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
    DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
}
```

```
    SYS_EnableGIE(4,0x3F);                  //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{
    if(DrvTIMER_GetIntFlag(E_TMA))
    {
        DrvTIMER_ClearIntFlag(E_TMA);           // Clear TMA interrupt flag
        MCUSTATUSbits.b_TMAdone=1;
        TimerA_count++;
    }

    if(DrvTIMER_GetIntFlag(E_TMB))
    {
        DrvTIMER_ClearIntFlag(E_TMB);           // Clear TMB interrupt flag
        MCUSTATUSbits.b_TMBdone=1;
        TimerB_count++;
    }

    if(DrvTIMER_GetIntFlag(E_TMC0))
    {
        DrvTIMER_ClearIntFlag(E_TMC0);          // Clear TMC0 interrupt flag
        MCUSTATUSbits.b_TMC0done=1;
        TimerC0_count++;
    }

    if(DrvTIMER_GetIntFlag(E_TMC1))
    {
        DrvTIMER_ClearIntFlag(E_TMC1);          // Clear TMC1 interrupt flag
        MCUSTATUSbits.b_TMC1done=1;
        TimerC1_count++;
```

```
    }
}


/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description    : CMP & OPA interrupt Service Routine (HW3).               */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description    : PT1 interrupt Service Routine (HW4).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description   : PT2 interrupt Service Routine (HW5).                      */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
/* Description   : Hardware I2C Initial.                                     */
/* Arguments     : None.                                                     */
```

```
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);      //Enable I2C interrupt and error interrupt
}

/*-------------------------------------------------------------------------*/
/* End Of File                                                             */
/*-------------------------------------------------------------------------*/
```

4. Digital IP(WDT)

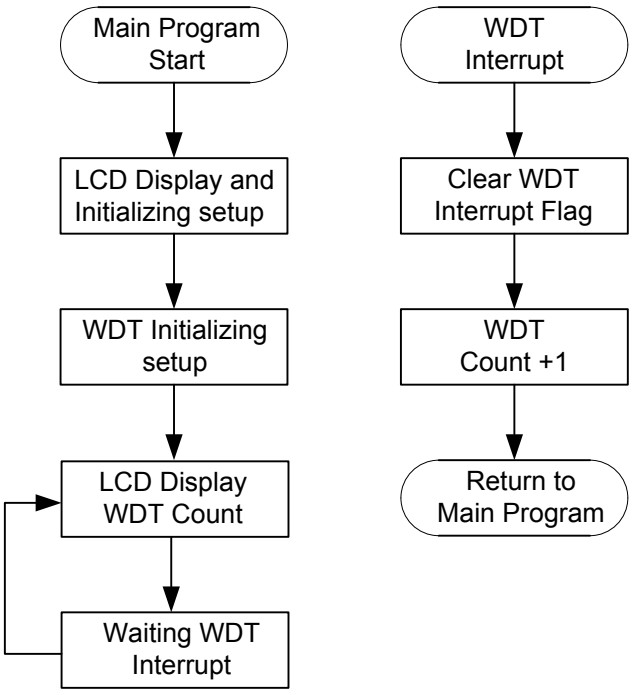
4.1. Example Name

HY16F188_WDT

4.2. Example Description

(1) WDT tutorial
(2) In this example code, set WDT initializing and WDT overflow condition, enable System GIE and wait WDT interrupt. WDT overflow can decide WDT interrupt frequency.
(3) Everytime WDT interrupt occurs, in the WDT interrupt service routine, variable "WDT Count" to do +1. And then return to main program to show "WDT Count" on LCD.

4.3. Software Flowchart



4.4. Program Description

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------------
 * Project Name : HY16F188_WDT
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *         GND    |
 *                |
 * ---------
 *******************************************************************************/
/*------------------------------------------------------------------------------*/
/* Includes                                                                     */
```

```
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvTimer.h"
#include "DrvI2C.h"
#include "DrvCLOCK.h"
#include "HY2613.h"
#include "my define.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char    _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_WDTdone:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define    I2CBufferSize    64


/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
```

```
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;
unsigned int WDT_count;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitaII2C(void);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

   WDT_count=0;
   DrvWDT_Open(E_IRQ,E_PRE_SCALER_D32);   //WDT IRQ open pre scaler 32
   DrvWDT_ClearWDT();                     //Clear WDT interrupt flag
   DrvTIMER_EnableInt(E_WDT);             //WDT interrupt enable

   InitaII2C();
   SYS_EnableGIE(4,0x3F);                 // Enable GIE(Global Interrupt)

   DisplayInit();
   ClearLCDframe();
   Delay(10000);
   DisplayHYcon();
   Delay(10000);

   while(1)                               //Wait for Interrupt
   {
      if(MCUSTATUSbits.b_WDTdone==1)
      {
         LCD_DATA_DISPLAY(WDT_count);
```

```
        MCUSTATUSbits.b_WDTdone=0;
      }
    }

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description    : I2C/UART/SPI interrupt Service Routine (HW0).            */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
   unsigned char I2C_Status,I2C_IntFlag;

   I2C_IntFlag=DrvI2C_ReadIntFlag();
   if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
   {
      I2C_Status=DrvI2C_GetStatusFlag();       //Get I2C Status Flag
      switch(I2C_Status)
      {
         case 0x90:   //MACTFlag+RWFlag
              {          //START has been transmitted
                 DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
                 DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
                 break;
              };
         case 0x84:   //MACTFlag+ACKFlag
              {          //Slave A + W has been transmitted. ACK has been received.
                 DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                 DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                 break;
              };
         case 0x80: //MACTFlag
              {         //Slave A + W has been transmitted. ACK has been received.
                 DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                 DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
```

```
            break;
        };
    case 0x30:
        {
            DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
            I2C_EndFlag=1;
            break;
        };
    case 0x8C:    //MACTFlag+DFFlag+ACKFlag
        {           //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
                DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
            }
            else
            {
                if(I2C_RW == I2C_WRITE)
                {
                    DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                    I2C_EndFlag=1;
                }
                else if(I2C_RW == I2C_READ)
                    DrvI2C_Ctrl(1,0,0,0);    //I2C as master sends START signal
                I2C_DataTxIndex=0;
            }
            break;
        };
    case 0x88:    //MACTFlag+DFFlag
        {           //DATA has been transmitted and NACK has been received
            DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
            I2C_EndFlag=1;
            break;
        };
    case 0xB0:
        {           //A repeated START has been transmitted.
            DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
```

```
            break;
        }
    case 0x94:    //MACTFlag+RWFlag+ACKFlag
        {           //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }else{
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {           //Data byte has been received. ACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            if((I2C_DataRxLen-1)>I2C_DataRxIndex)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }
            else
            {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {           //Data byte has been received. NACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
            I2C_EndFlag=1;
            break;
        };
    default:
        {
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            I2C_EndFlag=1;
            break;
        };
}
```

```
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
    DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
  }
  if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Error Interrupt Flag
  {
    I2C_EndFlag=1;
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
    DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
    DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
  }
  SYS_EnableGIE(4,0x3F);                //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description   : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{
  if(DrvTIMER_GetIntFlag (E_WDT))
  {
    WDT_count++;
    MCUSTATUSbits.b_WDTdone=1;
    DrvTIMER_ClearIntFlag(E_WDT);    //Clear WDT interrupt flag
  }
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description   : ADC interrupt Service Routine (HW2).                      */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
```

```
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{
```

```
}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
/* Description    : Hardware I2C Initial.                                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}
```

```
/*-------------------------------------------------------------------------*/
/* End Of File                                                             */
/*-------------------------------------------------------------------------*/
```

5. Digital IP(WDT Reset)

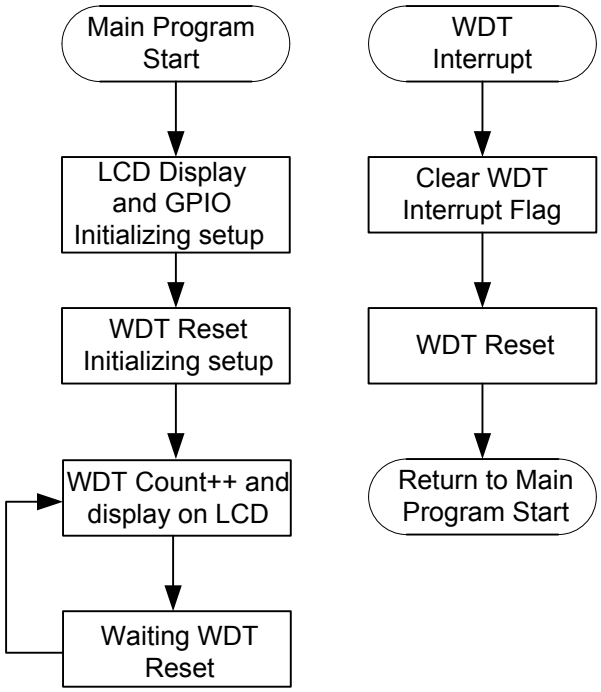
5.1. Example Name

HY16F188_WDT_Reset

5.2. Example Description

(1) WDT Reset tutorial
(2) In this example code, set WDT initializing and WDT overflow condition, enable System GIE and WDT Reset. Main program start to do variable "WDR Count" +1 and shows on LCD, wait WDT Reset occurrence.
(3) When "WDT Count" count up to 1500~2500, the WDT Reset will occur. Because each IC internal HAO frequency is different, so WDT Reset time is different for each IC.
(4) User can press PT1.7 button to clear WDT counter register. It also set WDT Count=0, restart to do variable "WDT Count" +1.

5.3. Software Flowchart



5.4. Program Description

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ------------------------------------------------------------------------
 * Project Name : HY16F188_WDT_Reset
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *         GND    |
 *                |
 * ---------
 *******************************************************************************/
/*-----------------------------------------------------------------------------*/
/* Includes                                                                    */
```

```
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvTimer.h"
#include "DrvI2C.h"
#include "DrvCLOCK.h"
#include "HY2613.h"
#include "my define.h"
#include "DrvGPIO.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char    _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_WDTdone:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;

typedef union _PTINTSTATUS
{
   char    _byte;
   struct
   {
      unsigned b_PTINT0done:1;
      unsigned b_PTINT1done:1;
      unsigned b_PTINT2done:1;
      unsigned b_PTINT3done:1;
      unsigned b_PTINT4done:1;
```

```
        unsigned b_PTINT5done:1;
        unsigned b_PTINT6Done:1;
        unsigned b_PTINT7Done:1;
    };
} PTINTSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define    I2CBufferSize    64
#define KEY_PORT E_PT1
#define KEYIN0 BIT6
#define KEYIN1 BIT7
#define KEYIN0_PIN 6
#define KEYIN1_PIN 7

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;
PTINTSTATUS    PT1INTSTATUSbits;
unsigned int WDT_count;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
```

```
int main(void)
{

  WDT_count=0;

  DrvWDT_Open(E_NMI,E_PRE_SCALER_D2048);                    //WDT IRQ open pre scaler 2048
  DrvWDT_ClearWDT();                                        //Clear WDT_count
  DrvWDT_ResetEnable();                                     //set WDNMI=1. 0x40108[6]=1b

  DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);             //clear PT1 interrupt flag
  DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_INPUT);          //set PT1.7/PT1.6 INPUT
  DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_PullHigh);       //enable PT1.7/PT1.6 pull high R
  DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_IntEnable);      //PT1_7/PT1_6 interrupt enable

  DrvGPIO_ClkGenerator(E_HS_CK,1);                          //Set IO sampling clock input source is HS_CK

  DrvGPIO_IntTrigger(KEY_PORT,KEYIN1|KEYIN0,E_N_Edge);      //PT1_7/1_6 interrupt trigger method is negative
edge
  MCUSTATUSbits._byte = 0;
  PT1INTSTATUSbits._byte = 0;


  InitalI2C();
  SYS_EnableGIE(4,0x3F);                                    // Enable GIE(Global Interrupt)

  DisplayInit();
  ClearLCDframe();
  Delay(10000);
  DisplayHYcon();
  Delay(10000);

  while(1)
  {
    for(WDT_count=0;WDT_count<999999;WDT_count++)
    {
      LCD_DATA_DISPLAY(WDT_count);                          //WDT Reset occur on about
WDT_count=1500~2500, it depends on the HAO frequency
      Delay(1000);
      if(PT1INTSTATUSbits.b_PTINT7Done)                     //if PT1.7 low
```

```
        {
          WDT_count=0;                                          //Set WDT_count=0,
          DrvWDT_ClearWDT();                                    //WDT re-count again, start from 0 to count
          PT1INTSTATUSbits.b_PTINT7Done=0;
        }
      }
    }
}
/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
  unsigned char I2C_Status,I2C_IntFlag;

  I2C_IntFlag=DrvI2C_ReadIntFlag();
  if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
  {
    I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
    switch(I2C_Status)
    {
      case 0x90:   //MACTFlag+RWFlag
          {          //START has been transmitted
            DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
            break;
          };
      case 0x84:   //MACTFlag+ACKFlag
          {          //Slave A + W has been transmitted. ACK has been received.
            DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;
          };
      case 0x80: //MACTFlag
          {         //Slave A + W has been transmitted. ACK has been received.
```

```
            DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
            break;
        };
    case 0x30:
        {
            DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
            I2C_EndFlag=1;
            break;
        };
    case 0x8C:   //MACTFlag+DFFlag+ACKFlag
        {           //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
                DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
            }
            else
            {
                if(I2C_RW == I2C_WRITE)
                {
                    DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
                    I2C_EndFlag=1;
                }
                else if(I2C_RW == I2C_READ)
                    DrvI2C_Ctrl(1,0,0,0);   //I2C as master sends START signal
                I2C_DataTxIndex=0;
            }
            break;
        };
    case 0x88:   //MACTFlag+DFFlag
        {           //DATA has been transmitted and NACK has been received
            DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
            I2C_EndFlag=1;
            break;
        };
    case 0xB0:
        {           //A repeated START has been transmitted.
```

```
            DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;
        }
    case 0x94:    //MACTFlag+RWFlag+ACKFlag
        {           //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }else{
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {           //Data byte has been received. ACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            if((I2C_DataRxLen-1)>I2C_DataRxIndex)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }
            else
            {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {           //Data byte has been received. NACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
            I2C_EndFlag=1;
            break;
        };
    default:
        {
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            I2C_EndFlag=1;
            break;
```

```
            };
        }
        DrvI2C_ClearIRQ();
        DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
        DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
    }
    if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
    {
        I2C_EndFlag=1;
        DrvI2C_ClearIRQ();
        DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
        DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
        DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
    }
    SYS_EnableGIE(4,0x3F);                  //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{
    if(DrvTIMER_GetIntFlag (E_WDT))
    {
        MCUSTATUSbits.b_WDTdone=1;
        DrvTIMER_ClearIntFlag(E_WDT);     //Clear WDT interrupt flag
    }
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
```

```
/* Remark          :                                                    */
/*---------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                            */
/* Description   : CMP & OPA interrupt Service Routine (HW3).          */
/* Arguments     : None.                                               */
/* Return Value : None.                                                */
/* Remark          :                                                   */
/*---------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                            */
/* Description   : PT1 interrupt Service Routine (HW4).                */
/* Arguments     : None.                                               */
/* Return Value : None.                                                */
/* Remark          :                                                   */
/*---------------------------------------------------------------------*/
void HW4_ISR(void)
{
   uint32_t PORT_IntFlag;

   PORT_IntFlag=DrvGPIO_GetIntFlag(KEY_PORT);
   if((PORT_IntFlag&KEYIN0)==KEYIN0)
   {
      PT1INTSTATUSbits.b_PTINT6Done=1;
   }
   if((PORT_IntFlag&KEYIN1)==KEYIN1)
   {
      PT1INTSTATUSbits.b_PTINT7Done=1;
   }
```

```
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);              //clear PT1_7/1_6 interrupt flag
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description    : PT2 interrupt Service Routine (HW5).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
    asm("nop"); //procedure define by customer.
    asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for(;num>0;num--)
    asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
```

```
/* Description    : Hardware I2C Initial.                                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                     */
/* Remark          :                                                        */
/*---------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                              */
/*---------------------------------------------------------------------------*/
```

6. Digital IP(RTC)

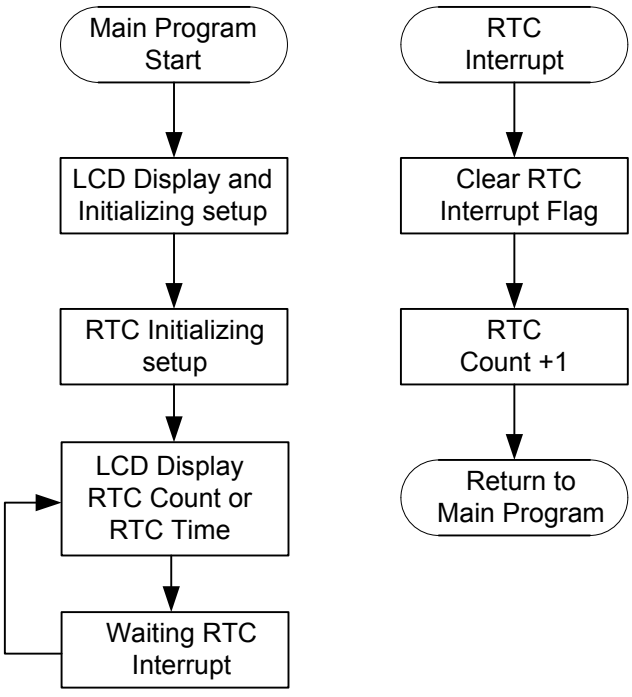
6.1. Example Name

HY16F188_RTC

6.2. Example Description

(1) RTC tutorial.
(2) Using #define to select to compile RTC_counter_show or TimerDtata_show.
(3) In this example code, set RTC initializing and RTC overflow condition, enable System GIE and wait RTC interrupt.
(4) If select RTC_counter_show, everytime RTC interrupt occurs, in the RTC interrupt service routine, variable "RTC Count" to do +1 and shows on LCD. If select TimerDtata_show, everytime RTC interrupt occurs, to do current time+1 and shows on LCD.

6.3. Software Flowchart



6.4. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------
 * Project Name : HY16F188_RTC
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *               |
 *       PT2.4 | LXIN
 *       PT2.5 | LXOUT
 *         GND    |
 *               |
 * ---------
 *****************************************************************************/
```

```
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvI2C.h"
#include "DrvCLOCK.h"
#include "DrvRTC.h"
#include "HY2613.h"
#include "my define.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define   I2CBufferSize   64
//#define   RTC_counter_show
#define   TimerDtata_show
```

```
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;
unsigned int sec,min,hour,week,day,month,year ;
unsigned int RTC_counter;
unsigned int TimerDtata;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);
void InitalRTC(void);
int ComputeWeek(int TempYear, int TempMonth, int TempDay);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

   S_DRVRTC_TIME_DATA_T sCurTime;            //Setting Start
   RTC_counter=0;
   TimerDtata=0;

   InitalI2C();

   SYS_EnableGIE(4,0x3F);                    // Enable GIE(Global Interrupt)

   DisplayInit();
   ClearLCDframe();
   Delay(10000);
   DisplayHYcon();
```

```
    Delay(10000);
    MCUSTATUSbits._byte = 0;

    InitalRTC();


    while(1)
    {
      if(MCUSTATUSbits.b_RTCdone==1)
      {
        DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
        TimerDtata=sCurTime.u32cSecond+sCurTime.u32cMinute*100+sCurTime.u32cHour*10000;
#if defined(TimerDtata_show)
        LCD_DATA_DISPLAY(TimerDtata);
#endif
#if defined(RTC_counter_show)
        LCD_DATA_DISPLAY(RTC_counter);
#endif

        sec=sCurTime.u32cSecond;
        min=sCurTime.u32cMinute;
        hour=sCurTime.u32cHour;
        week=sCurTime.u32cDayOfWeek;
        day=sCurTime.u32cDay;
        month=sCurTime.u32cMonth;
        year=sCurTime.u32Year;
        MCUSTATUSbits.b_RTCdone=0;
      }
    }

    return 0;

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
```

```
/* Remark            :                                                                    */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
   unsigned char I2C_Status,I2C_IntFlag;

   I2C_IntFlag=DrvI2C_ReadIntFlag();
   if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
   {
      I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
      switch(I2C_Status)
      {
         case 0x90:   //MACTFlag+RWFlag
               {          //START has been transmitted
                  DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
                  DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
                  break;
               };
         case 0x84:   //MACTFlag+ACKFlag
               {          //Slave A + W has been transmitted. ACK has been received.
                  DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                  DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                  break;
               };
         case 0x80: //MACTFlag
               {         //Slave A + W has been transmitted. ACK has been received.
                  DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                  DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                  break;
               };
         case 0x30:
               {
                  DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                  I2C_EndFlag=1;
                  break;
               };
         case 0x8C:   //MACTFlag+DFFlag+ACKFlag
               {          //DATA has been transmitted and ACK has been received
                  if(I2C_DataTxIndex<I2C_DataTxLen)
```

```
        {
          DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
          DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
        }
        else
        {
          if(I2C_RW == I2C_WRITE)
          {
            DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
            I2C_EndFlag=1;
          }
          else if(I2C_RW == I2C_READ)
            DrvI2C_Ctrl(1,0,0,0);   //I2C as master sends START signal
          I2C_DataTxIndex=0;
        }
        break;
      };
case 0x88:   //MACTFlag+DFFlag
      {          //DATA has been transmitted and NACK has been received
        DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
        I2C_DataTxIndex=0;
        I2C_EndFlag=1;
        break;
      };
case 0xB0:
      {          //A repeated START has been transmitted.
        DrvI2C_WriteData(I2C_TARGET | I2C_READ);   //Send Slave Address & R/W Bit
        DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
        break;
      }
case 0x94:   //MACTFlag+RWFlag+ACKFlag
      {          //Slave A + R has been transmitted. ACK has been received.
        if(I2C_DataRxLen>1)
        {
          DrvI2C_Ctrl(0,0,0,1);   //Set ACK bit
        }else{
          DrvI2C_Ctrl(0,0,0,0);   //Clear all I2C flag
        }
        break;
```

```
            };
        case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
            {           //Data byte has been received. ACK has been transmitted.
              I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
              if((I2C_DataRxLen-1)>I2C_DataRxIndex)
              {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
              }
              else
              {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
              }
              break;
            };
        case 0x98:    //MACTFlag+RWFlag+DFFlag
            {           //Data byte has been received. NACK has been transmitted.
              I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
              DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
              I2C_EndFlag=1;
              break;
            };
        default:
            {
              DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
              I2C_EndFlag=1;
              break;
            };
    }
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
    DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
}
if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
{
    I2C_EndFlag=1;
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
    DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
    DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
```

```
    }
    SYS_EnableGIE(4,0x3F);                  //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{
    if(DrvRTC_ReadState()&&0x2)    //check PTF flag
    {
        DrvRTC_ClearState(E_DRVRTC_CLEAR_ALL);
        DrvRTC_ClearIntFlag();
        RTC_counter++;
        MCUSTATUSbits.b_RTCdone=1;
    }
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description    : CMP & OPA interrupt Service Routine (HW3).               */
/* Arguments      : None.                                                    */
```

```
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                */
/* Description  : PT1 interrupt Service Routine (HW4).                     */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                */
/* Description  : PT2 interrupt Service Routine (HW5).                     */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                  */
/* Description  : Exception Service Routines.                              */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
```

```
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
/* Description    : Hardware I2C Initial.                                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);      //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);         //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalRTC()                                                */
/* Description    : RTC Initialization Subroutines .                         */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalRTC()
{
```

```
    S_DRVRTC_TIME_DATA_T sCurTime;              //Setting Start
    DrvRTC_ClkConfig(1);
    DrvCLOCK_EnableLowOSC(E_EXTERNAL,130000);
    DrvRTC_ClockSource(0);                      //RTC clock source =LSXT

    //DRVRTC_CURRENT_TIME or Alarm Time
    DrvRTC_WriteEnable();
    DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
    sCurTime.u8cClockDisplay=0;                 //DRVRTC_CLOCK_12//DRVRTC_CLOCK_24
    sCurTime.u8cAmPm=0;                         //DRVRTC_AM//DRVRTC_PM
    sCurTime.u32cSecond=40;
    sCurTime.u32cMinute=59;
    sCurTime.u32cHour=23;
    sCurTime.u32cDay=18;
    sCurTime.u32cMonth=2;
    sCurTime.u32Year=2018;
    sCurTime.u32cDayOfWeek=ComputeWeek(sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay);
    sCurTime.u8IsEnableWakeUp=0;                //WK
    DrvRTC_Write(DRVRTC_CURRENT_TIME,&sCurTime);
    DrvRTC_HourFormat(E_DRVRTC_HOUR_24);    //1:12hour 0:24hour
    DrvRTC_Enable();
    DrvRTC_PeriodicTimeEnable(E_DRVRTC_1_8_SEC);
    DrvRTC_ClearIntFlag();
    DrvRTC_EnableInt();                         //Enable RTC interrupt
}

/*---------------------------------------------------------------------------*/
/* Function Name: ComputeWeek(int TempYear, int TempMonth, int TempDay)      */
/* Description    : Compute Week Subroutines.                                */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
int ComputeWeek(int TempYear, int TempMonth, int TempDay)
{
    int TempWeek;
    if (TempMonth >= 3)
    {
        TempMonth = TempMonth - 2;
```

```
    }
    else
    {
        TempMonth = TempMonth + 10;
        TempYear--;
    }

    TempWeek = TempYear + (int)(TempYear / 4) - (int)(TempYear / 100) +(int)(TempYear / 400) + (int)(2.6 * TempMonth -
0.2) + TempDay;
    TempWeek = TempWeek - 7*(int)(TempWeek / 7);
    return (TempWeek);
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

7. Digital IP(PWM)

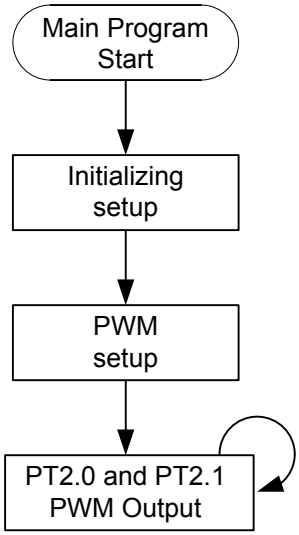
7.1. Example Name

HY16F188_PWM

7.2. Example Description

(1) PWM tutorial.
(2) Set HAO=4MHz and TimerB overflow condition.
(3) Set PWM register and PWM Duty, setting PWM output I/O port is output mode.
(4) PT2.0 is PWM0 output, PT2.1 is PWM1 output.

7.3. Software Flowchart



7.4. Program Description

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ------------------------------------------------------------------------
 * Project Name : HY16F188_PWM
 * IDE tooling   : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.   : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device    :
 * Description   :
 * Created Date : 2018/2/18
 * Created by    :
 *
 * Program Description:
 * --------------------
 *
 *******************************************************************************/
/*------------------------------------------------------------------------------*/
/* Includes                                                                     */
/*------------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvREG32.h"
#include "DrvTimer.h"
#include "DrvCLOCK.h"
```

```
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/



/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);


/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
   DrvCLOCK_SelectIHOSC(1);                        // Select HAO 4MHz
   DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
   SYS_DisableGIE();                              //Disable GIE

   DrvPWM0_Open(0,1,4);                            //PWM0 Enable Port 2.0 =PWMO0, Port 2.1 =PWMO1(Set
PWM0=PT2.0)
   DrvPWM1_Open(1,1,4);                            //PWM1 Enable Port 2.0 =PWMO0, Port 2.1 =PWMO1(Set
PWM1=PT2.1)
   DrvPWM_CountCondition(0x7fff,0x3fff);          //PWM0 Duty 0X7FFF, PWM0=PT2.0
                                                   //PWM1 Duty 0X3FFF, PWM1=PT2.1
   DrvGPIO_Open(E_PT2,0x01|0x02,E_IO_OUTPUT);      //PT2.0, PT2.1 Set Output
```

```
    DrvTMBC_Clk_Source(0,0);                              //TMB Clock Enable/1
    DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0xffff);    //TMB Overflow 0XFFFF
                                                          //PWM Period 0XFFFF
    while(1);
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description    : I2C/UART/SPI interrupt Service Routine (HW0).            */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{



}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1). */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
```

```
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{


}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

8. Digital IP(Flash)

8.1. Example Name

HY16F188_Flash

8.2. Example Description

(1) Flash burn and read tutorial.
(2) In this example code, first, execute flash burn page and read out and to do verify, if burn page content and read out content is different, program stuck in while(1).
(3) Second, execute flash burn word and read out to do verify, if burn page content and read out content is different, program stuck in while(1).
Note1 : User has to do SYS_DisableGIE, before execute Flash burn/read function. Disable system global GIE function, it can prevent program exception when executing Flash burn/read function.
Note2 : VDD3V have to more than 2.7V, it can prevent program burn error when executing Flash burn function.

8.3. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------------
 * Project Name : HY16F188_Flash
 * IDE tooling   : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.   : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
```

```
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 ********************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "DrvCLOCK.h"
#include "System.h"
#include "DrvI2C.h"
#include "HY2613.h"
#include "my define.h"
#include "DrvFlash.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);


/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
```

```
/*---------------------------------------------------------------------------*/
int main(void)
{
   int index,error;
   int BufferTx[32];
   int BufferRx[32];

   SYS_DisableGIE();                    //before execute Flash burn, must be to do that Disable GIE

   for(index=0;index<32;index++)
   {
      BufferTx[index]=index;            //to set BufferTx[0]=0x0....BufferTx[31]=0x1f
      BufferRx[index]=0xFFFFFFFF;       //to set BufferRx[0]=0xFFFFFFFF....BufferRx[31]=0xFFFFFFFF
   }
   ROM_BurnPage(0x8000,0x2000,BufferTx);
   ReadPage(0x8000,BufferRx);           //Read BufferRx to make sure BurnPage data

   for(index=0;index<32;index++)
   {
      BufferTx[index]=31-index;         //to set BufferTx[0]=0x1f....BufferTx[31]=0x0
      BufferRx[index]=0xFFFFFFFF;       //to set BufferRx[0]=0xFFFFFFFF....BufferRx[31]=0xFFFFFFFF
   }
   ROM_BurnPage(0x8000,0x2000,BufferTx);
   ReadPage(0x8000,BufferRx);           //Read BufferRx to make sure BurnPage data

   //verify the ROM_BurnPage function, if fail to stuck in the while(1)
   for(index=0;index<32;index++)
   {
      if(BufferTx[index]!=BufferRx[index])
      {
          while(1);
      }

   }

   error=DrvFlash_Burn_Word(0x8014,0x2000,0x12345678);
   if(error==0)                         //if Error=0, it means BurnWord success
      ReadPage(0x8014,BufferRx);
```

```
    error=DrvFlash_Burn_Word(0x8010,0x2000,0x12345678);
    if(error==0)                          //if Error=0, it means BurnWord success
      ReadPage(0x8010,BufferRx);

    error=DrvFlash_Burn_Word(0x8020,0x2000,0x12345678);
    if(error==0)                          //if Error=0, it means BurnWord success
      ReadPage(0x8020,BufferRx);

    while(1);
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description   : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description   : ADC interrupt Service Routine (HW2).                      */
```

```
/* Arguments      : None.                                                    */
/* Return Value : None.                                                    */
/* Remark         :                                                        */
/*-------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                */
/* Description   : CMP & OPA interrupt Service Routine (HW3).              */
/* Arguments      : None.                                                  */
/* Return Value : None.                                                    */
/* Remark         :                                                        */
/*-------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                */
/* Description   : PT1 interrupt Service Routine (HW4).                    */
/* Arguments      : None.                                                  */
/* Return Value : None.                                                    */
/* Remark         :                                                        */
/*-------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                */
/* Description   : PT2 interrupt Service Routine (HW5).                    */
/* Arguments      : None.                                                  */
/* Return Value : None.                                                    */
/* Remark         :                                                        */
```

```
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

9. Digital IP(GPIO)

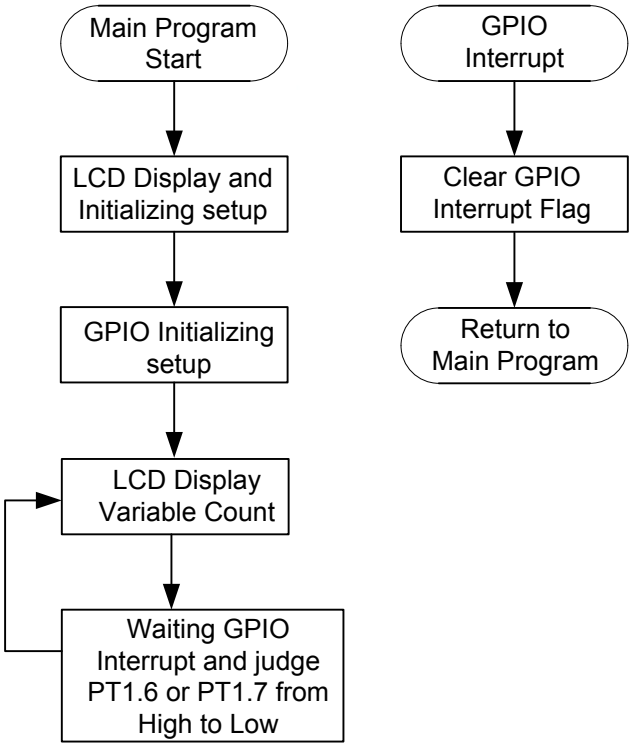
9.1. Example Name

HY16F188_GPIO

9.2. Example Description

(1) GPIO tutorial
(2) Setup GPIO and LCD initializing.
(3) If press button PT1.6, to do variable count +1 and LCD display count.
(4) If press button PT1.7, to do variable count -1 and LCD display count.

9.3. Software Flowchart



9.4. Program Description

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ------------------------------------------------------------------------
 * Project Name : HY16F188_GPIO
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *         GND    |
 *                |
 * ---------
 *******************************************************************************/
/*-------------------------------------------------------------------------*/
/* Includes                                                                */
```

```
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvI2C.h"
#include "DrvCLOCK.h"
#include "HY2613.h"
#include "my define.h"
#include "DrvGPIO.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char    _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_WDTdone:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;

typedef union _PTINTSTATUS
{
   char    _byte;
   struct
   {
      unsigned b_PTINT0done:1;
      unsigned b_PTINT1done:1;
      unsigned b_PTINT2done:1;
      unsigned b_PTINT3done:1;
      unsigned b_PTINT4done:1;
      unsigned b_PTINT5done:1;
```

```
        unsigned b_PTINT6Done:1;
        unsigned b_PTINT7Done:1;
    };
} PTINTSTATUS;



/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define    I2CBufferSize    64
#define KEY_PORT E_PT1
#define KEYIN0 BIT6
#define KEYIN1 BIT7
#define KEYIN0_PIN 6
#define KEYIN1_PIN 7

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;
PTINTSTATUS    PT1INTSTATUSbits;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
```

```
    int count=0;

    DrvGPIO_ClkGenerator(E_HS_CK,1);                          //Set IO sampling clock input source is HS_CK
    DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_INPUT);          //set PT1.2/PT1.1 INPUT
    DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_PullHigh);       //enable PT1.2/PT1.1 pull high R
    DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_IntEnable);      //PT1.2/PT1.1 interrupt enable
    DrvGPIO_IntTrigger(KEY_PORT,KEYIN1|KEYIN0,E_N_Edge);      //PT1.2/PT1.1 interrupt trigger method is negative
edge
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);             //clear PT1 interrupt flag
    MCUSTATUSbits._byte = 0;
    PT1INTSTATUSbits._byte = 0;


    InitalI2C();
    SYS_EnableGIE(4,0x3F);                                    // Enable GIE(Global Interrupt)
    DisplayInit();
    ClearLCDframe();
    LCD_DATA_DISPLAY(count);

    while(1)
    {
      if(PT1INTSTATUSbits.b_PTINT6Done)                       //if PT1.6 low
      {
        LCD_DATA_DISPLAY(count++);
        PT1INTSTATUSbits.b_PTINT6Done=0;
      }
      if(PT1INTSTATUSbits.b_PTINT7Done)                       //if PT1.7 low
      {
        LCD_DATA_DISPLAY(count--);
        PT1INTSTATUSbits.b_PTINT7Done=0;
      }
    }

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
```

```
/* Arguments      : None.                                                    */
/* Return Value : None.                                                    */
/* Remark          :                                                       */
/*-------------------------------------------------------------------------*/
void HW0_ISR(void)
{
   unsigned char I2C_Status,I2C_IntFlag;

   I2C_IntFlag=DrvI2C_ReadIntFlag();
   if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
   {
      I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
      switch(I2C_Status)
      {
         case 0x90:   //MACTFlag+RWFlag
               {          //START has been transmitted
                  DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
                  DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
                  break;
               };
         case 0x84:   //MACTFlag+ACKFlag
               {          //Slave A + W has been transmitted. ACK has been received.
                  DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                  DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                  break;
               };
         case 0x80: //MACTFlag
               {         //Slave A + W has been transmitted. ACK has been received.
                  DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                  DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                  break;
               };
         case 0x30:
               {
                  DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                  I2C_EndFlag=1;
                  break;
               };
         case 0x8C:   //MACTFlag+DFFlag+ACKFlag
```

```
    {           //DATA has been transmitted and ACK has been received
      if(I2C_DataTxIndex<I2C_DataTxLen)
      {
        DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
        DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
      }
      else
      {
        if(I2C_RW == I2C_WRITE)
        {
          DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
          I2C_EndFlag=1;
        }
        else if(I2C_RW == I2C_READ)
          DrvI2C_Ctrl(1,0,0,0);   //I2C as master sends START signal
        I2C_DataTxIndex=0;
      }
      break;
    };
  case 0x88:   //MACTFlag+DFFlag
    {           //DATA has been transmitted and NACK has been received
      DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
      I2C_DataTxIndex=0;
      I2C_EndFlag=1;
      break;
    };
  case 0xB0:
    {           //A repeated START has been transmitted.
      DrvI2C_WriteData(I2C_TARGET | I2C_READ);   //Send Slave Address & R/W Bit
      DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
      break;
    }
  case 0x94:   //MACTFlag+RWFlag+ACKFlag
    {           //Slave A + R has been transmitted. ACK has been received.
      if(I2C_DataRxLen>1)
      {
        DrvI2C_Ctrl(0,0,0,1);   //Set ACK bit
      }else{
        DrvI2C_Ctrl(0,0,0,0);   //Clear all I2C flag
```

```
            }
            break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {           //Data byte has been received. ACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            if((I2C_DataRxLen-1)>I2C_DataRxIndex)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }
            else
            {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {           //Data byte has been received. NACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
            I2C_EndFlag=1;
            break;
        };
    default:
        {
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            I2C_EndFlag=1;
            break;
        };
  }
  DrvI2C_ClearIRQ();
  DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
  DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
}
if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
{
  I2C_EndFlag=1;
  DrvI2C_ClearIRQ();
  DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
```

```
        DrvI2C_ClearIntFlag(1);          //Clear I2C Interrupt Flag(I2CEIF)
        DrvI2C_Ctrl(0,0,0,0);            //Clear all I2C flag
    }
    SYS_EnableGIE(4,0x3F);               //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description   : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description   : ADC interrupt Service Routine (HW2).                      */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description   : CMP & OPA interrupt Service Routine (HW3).                */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
```

```
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description    : PT1 interrupt Service Routine (HW4).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{
   uint32_t PORT_IntFlag;

   PORT_IntFlag=DrvGPIO_GetIntFlag(KEY_PORT);
   if((PORT_IntFlag&KEYIN0)==KEYIN0)
   {
      PT1INTSTATUSbits.b_PTINT6Done=1;
   }
   if((PORT_IntFlag&KEYIN1)==KEYIN1)
   {
      PT1INTSTATUSbits.b_PTINT7Done=1;
   }
   DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);          //clear PT1_7/1_6 interrupt flag
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description    : PT2 interrupt Service Routine (HW5).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{


}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
/* Description    : Hardware I2C Initial.                                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);      //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);         //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
```

```
/*-------------------------------------------------------------------------*/
```

10. Analog IP(8 bit Resistance Ladder DAC)

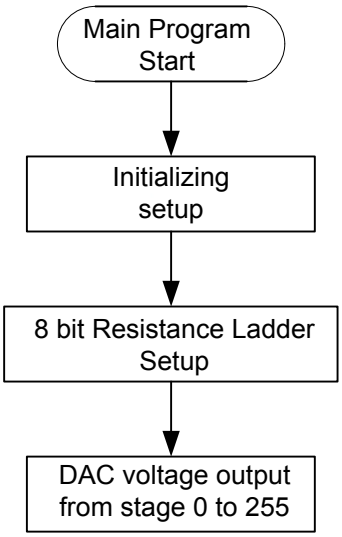
10.1. Example Name

HY16F188_DAC

10.2. Example Description

(1) 8 bit Resistance ladder DAC tutorial
(2) In this example code, first, enable VDDA, and 8 bit Resistance Ladder positive set as VDDA, negative set as VSS. In this setting, the maximum DAC output is equal to VDDA.
(3) DAC voltage output from stage 0 to 255, user can measure or observe DAC voltage output from IC pin DAO.

10.3. Software Flowchart



10.4. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------
 * Project Name : HY16F188_DAC
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "DrvREG32.h"
#include "DrvPMU.h"
#include "DrvDAC.h"
#include "my define.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
```

```
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);


/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  unsigned char i;

  DrvPMU_VDDA_LDO_Ctrl(E_LDO);                  //LDO ON
  DrvPMU_VDDA_Voltage(E_VDDA2_4);               //VDDA=2.4

  DrvDAC_Open(E_DAC_PVDDA,E_DAC_NVSSA,0);       //DAC_Vrefp=VDDA, DAC_Vrefn= VSSA, DAO=0
  DrvDAC_SetoutputIO(ENABLE);                    //DAC output with PT3.1
  DrvDAC_EnableOutput();                        //DAC output enable
  DrvDAC_Enable();                              //DAC IP enable

  while(1)
  {
    for(i=0;i<256;i++)
    {
      DrvDAC_DABIT(i);
      Delay(1000);
      //User can use meter to check the pin49(PT3.1) to pin100(VSS) to check the DAC output voltage
    }
```

```
    }
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description  : I2C/UART/SPI interrupt Service Routine (HW0).              */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description  : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).   */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description  : ADC interrupt Service Routine (HW2).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
```

```
/* Description    : Exception Service Routines.                    */
/* Arguments      : None.                                          */
/* Return Value : None.                                            */
/* Remark         :                                                */
/*-----------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*-----------------------------------------------------------------*/
/* Software Delay Subroutines                                      */
/*-----------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*-----------------------------------------------------------------*/
/* End Of File                                                     */
/*-----------------------------------------------------------------*/
```

11. Analog IP(OPA)

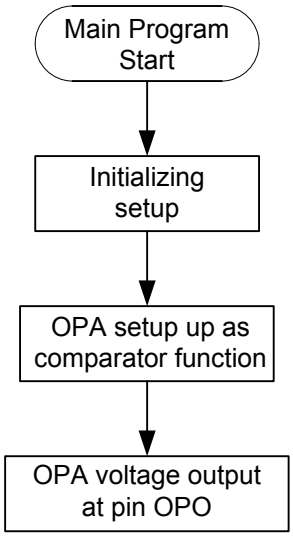
11.1. Example Name

HY16F188_OPA

11.2. Example Description

(1) OPAMP tutorial
(2) Enable analog power REFO=1.2V
(3) OPAMP positive set as REFO , OPAMP negative set as AIO5.
(4) OPAMP set as comparator function. When REFO voltage more than AIO5, OPO pin(PT3.7) output high, when REFO voltage less than AIO5, OPO pin(PT3.7) output low.
(5) Everytime OPAMP interrupt occur, PT2.0 will do high or low output status changed.

11.3. Software Flowchart



11.4. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------
 * Project Name : HY16F188_OPA
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "DrvGPIO.h"
#include "DrvOP.h"
#include "DrvPMU.h"
#include "DrvREG32.h"
#include "HY16F188.h"
#include "System.h"
#include "my define.h"
```

```
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
      unsigned b_OPAdone:1;
   };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define KEY_PORT E_PT2
#define KEYIN0 BIT0

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS   MCUSTATUSbits;
unsigned char i;

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);

/*---------------------------------------------------------------------------*/
```

```
/* Main Function                                                                    */
/*--------------------------------------------------------------------------------*/
int main(void)
{
   i=0;

   DrvPMU_VDDA_LDO_Ctrl(E_LDO);                    //LDO ON
   DrvPMU_VDDA_Voltage(E_VDDA2_4);                 //VDDA=2.4
   DrvPMU_REFO_Enable();                          //REFO ON

   DrvGPIO_Open(E_PT2,KEYIN0,E_IO_OUTPUT);     //PT2.0 Output

   DrvOP_Open();
   DrvOP_PInput(0x08);                          //OPA positive reference input selection REFO
   DrvOP_NInput(0x02);                          //OPA negative reference input selection AIO5
   DrvOP_OPOoutEnable();                         //OPA Out Enable PT3.7. If REFO>AIO5, PT3.7=High, IF REFO<AIO5,
PT3.7=Low
   DrvOP_OPDEN(1);                               //OPDEN=1b

   DrvOP_EnableInt();
   MCUSTATUSbits._byte = 0;
   SYS_EnableGIE(4,0x3F);                       //Enable GIE(Global Interrupt)

   while(1)
   {
       //If REFO>AIO5, enter HW3_ISR()
   }

   return 0;
}

/*--------------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                       */
/* Description    : I2C/UART/SPI interrupt Service Routine (HW0).                 */
/* Arguments      : None.                                                         */
/* Return Value : None.                                                           */
/* Remark         :                                                               */
/*--------------------------------------------------------------------------------*/
void HW0_ISR(void)
```

```
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description    : CMP & OPA interrupt Service Routine (HW3).               */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{
  if(DrvOP_ReadIntFlag())
  {
```

```
        if(i==ENABLE)
        {
           DrvGPIO_ClrPortBits(E_PT2,KEYIN0);    //PT2.0 Output Low
           i=DISABLE;
        }
         else
        {
           DrvGPIO_SetPortBits(E_PT2,KEYIN0);    //PT2.0 Output High
           i=ENABLE;
        }
           DrvOP_ClearIntFlag();                 //Clear OPA interrupt flag
     }
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description    : PT1 interrupt Service Routine (HW4).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description    : PT2 interrupt Service Routine (HW5).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
```

```
/* Function Name: tlb_exception_handler()                                  */
/* Description    : Exception Service Routines.                            */
/* Arguments      : None.                                                  */
/* Return Value : None.                                                    */
/* Remark         :                                                        */
/*-------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*-------------------------------------------------------------------------*/
/* Software Delay Subroutines                                              */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}


/*-------------------------------------------------------------------------*/
/* End Of File                                                             */
/*-------------------------------------------------------------------------*/
```

12. Analog IP(ADC)

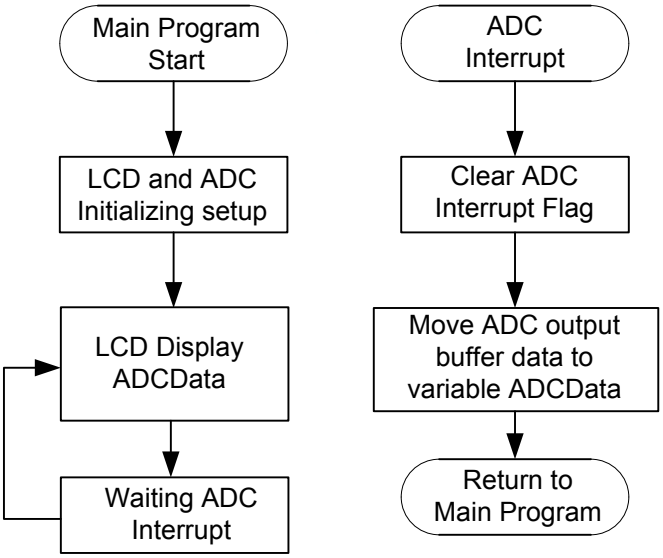
12.1. Example Name

HY16F188_ADC

12.2. Example Description

(1) ADC tutorial. Implement ADC interrupt function and capture ADC output buffer data.
(2) ADC Initializing, ADC analog power set as VDDA,ADC OSR=32768, ADC output rate=10Hz, ADC input channel set as AIO0-AIO1, ADC reference voltage set as VDDA-VSS.
(3) After ADC Initializing, Enable System GIE and wait ADC interrupt. ADC OSR can decide the ADC interrupt frequency.
(4) LCD Display variable "ADCData"

12.3. Software Flowchart



12.4. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------------
 * Project Name : HY16F188_ADC
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *               |
 *        AIO0 | ADC Input +
 *        AIO1 | ADC Input -
 *        GND    |
 *               |
 * ---------
```

```
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvI2C.h"
#include "DrvADC.h"
#include "DrvPMU.h"
#include "DrvClock.h"
#include "HY2613.h"
#include "my define.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define   I2CBufferSize   64
//#define HAO_2MHZ
//#define HAO_4MHZ
```

```
#define HAO_10MHZ



/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;
int ADCData;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);
void InitalADC(void);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

   InitalI2C();
   InitalADC();
   SYS_EnableGIE(4,0x3F);                    // Enable GIE(Global Interrupt)

   DisplayInit();
   ClearLCDframe();
   Delay(10000);
   DisplayHYcon();
   Delay(10000);
   MCUSTATUSbits._byte = 0;

   while(1)
```

```
    {
        if(MCUSTATUSbits.b_ADCdone)
        {
            LCD_DATA_DISPLAY(ADCData>>16);        //give up 16bits.
            MCUSTATUSbits.b_ADCdone=0;
        }
    }
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
    unsigned char I2C_Status,I2C_IntFlag;

    I2C_IntFlag=DrvI2C_ReadIntFlag();
    if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Interrupt Flag
    {
        I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
        switch(I2C_Status)
        {
            case 0x90:    //MACTFlag+RWFlag
                {          //START has been transmitted
                    DrvI2C_WriteData(I2C_TARGET);    //Send Slave Address & R/W Bit
                    DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
                    break;
                };
            case 0x84:    //MACTFlag+ACKFlag
                {          //Slave A + W has been transmitted. ACK has been received.
                    DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                    DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
                    break;
                };
```

```
case 0x80: //MACTFlag
    {       //Slave A + W has been transmitted. ACK has been received.
      DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
      DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
      break;
    };
case 0x30:
    {
      DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
      I2C_EndFlag=1;
      break;
    };
case 0x8C:   //MACTFlag+DFFlag+ACKFlag
    {         //DATA has been transmitted and ACK has been received
      if(I2C_DataTxIndex<I2C_DataTxLen)
      {
        DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
        DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
      }
      else
      {
        if(I2C_RW == I2C_WRITE)
        {
          DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
          I2C_EndFlag=1;
        }
        else if(I2C_RW == I2C_READ)
          DrvI2C_Ctrl(1,0,0,0);   //I2C as master sends START signal
        I2C_DataTxIndex=0;
      }
      break;
    };
case 0x88:   //MACTFlag+DFFlag
    {         //DATA has been transmitted and NACK has been received
      DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
      I2C_DataTxIndex=0;
      I2C_EndFlag=1;
      break;
    };
```

```
case 0xB0:
    {       //A repeated START has been transmitted.
      DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
      DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
      break;
    }
case 0x94:    //MACTFlag+RWFlag+ACKFlag
    {       //Slave A + R has been transmitted. ACK has been received.
      if(I2C_DataRxLen>1)
      {
        DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
      }else{
        DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
      }
      break;
    };
case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
    {       //Data byte has been received. ACK has been transmitted.
      I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
      if((I2C_DataRxLen-1)>I2C_DataRxIndex)
      {
        DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
      }
      else
      {
        DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
      }
      break;
    };
case 0x98:    //MACTFlag+RWFlag+DFFlag
    {       //Data byte has been received. NACK has been transmitted.
      I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
      DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
      I2C_EndFlag=1;
      break;
    };
default:
    {
      DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
```

```
                I2C_EndFlag=1;
                break;
            };
        }
        DrvI2C_ClearIRQ();
        DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
        DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
    }
    if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
    {
        I2C_EndFlag=1;
        DrvI2C_ClearIRQ();
        DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
        DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
        DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
    }
    SYS_EnableGIE(4,0x3F);                  //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
```

```
void HW2_ISR(void)
{
  if(DrvADC_ReadIntFlag())
  {
    DrvADC_ClearIntFlag();
    ADCData=DrvADC_GetConversionData();
    MCUSTATUSbits.b_ADCdone=1;
  }
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
```

```
/* Remark            :                                                    */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                  */
/* Description    : Exception Service Routines.                            */
/* Arguments      : None.                                                  */
/* Return Value : None.                                                    */
/* Remark            :                                                     */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                              */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitaII2C(void)                                          */
/* Description    : Hardware I2C Initial.                                  */
/* Arguments      : None.                                                  */
/* Return Value : None.                                                    */
/* Remark            :                                                     */
/*---------------------------------------------------------------------------*/
void InitaII2C(void)
{
   DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
```

```
#if defined(HAO_2MHZ)
   DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
#endif
#if defined(HAO_4MHZ)
   DrvI2C_Open(0x9);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 4MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 4000000/[4*(9+1)]=100kHz
#endif
#if defined(HAO_10MHZ)
   DrvI2C_Open(0x18);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 10MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 10000000/[4*(24+1)]=100kHz
#endif

   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}



/*---------------------------------------------------------------------------*/
/* Function Name: InitalADC()                                                */
/* Description    : ADC Initialization Subroutines                           */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void InitalADC(void)
{

#if defined(HAO_2MHZ)
   DrvCLOCK_EnableHighOSC(E_INTERNAL,10);     //Select HSRC
   DrvCLOCK_SelectIHOSC(0);                   //Select internal 2MHZ
   DrvADC_ClkEnable(0,1);                     //Setting ADC CLOCK ADCK=HS_CK/6 & Rising edge is high
#endif
#if defined(HAO_4MHZ)
   DrvCLOCK_EnableHighOSC(E_INTERNAL,10);     //Select HSRC
   DrvCLOCK_SelectIHOSC(1);                   //Select internal 4MHZ
   DrvADC_ClkEnable(1,1);                     //Setting ADC CLOCK ADCK=HS_CK/12 & Rising edge is high
#endif
#if defined(HAO_10MHZ)
```

```
    DrvCLOCK_EnableHighOSC(E_INTERNAL,10);     //Select HSRC
    DrvCLOCK_SelectIHOSC(2);                   //Select internal 10MHZ
    DrvADC_ClkEnable(2,1);                     //Setting ADC CLOCK ADCK=HS_CK/30 & Rising edge is high
#endif

    //Set VDDA voltage
    DrvPMU_VDDA_LDO_Ctrl(E_LDO);
    DrvPMU_VDDA_Voltage(E_VDDA2_4);
    DrvPMU_BandgapEnable();

    Delay(0x1000);
    DrvPMU_AnalogGround(ENABLE);               //ADC analog ground source selection.
                                               //1 : Enable buffer and use internal source(need to work with ADC)

    //Set ADC input pin
    DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1); //Set the ADC positive/negative input voltage
source.
    DrvADC_InputSwitch(OPEN);                  //ADC signal input (positive and negative) short(VISHR) control.
    DrvADC_RefInputShort(OPEN);                //Set the ADC reference input (positive and negative) short(VRSHR)
control.

    DrvADC_Gain(ADC_PGA_Disable,ADC_PGA_Disable); //Input signal gain for modulator.
    DrvADC_DCoffset(0);                        //DC offset input voltage selection (VREF=REFP-REFN)
    DrvADC_RefVoltage(0,0);                    //Set the ADC reference voltage. VDDA-VSSA
    DrvADC_FullRefRange(1);                    //Set the ADC full reference range select.
                                               //0: Full reference range input
                                               //1: 1/2 reference range input
    DrvADC_OSR(0);                             //0 : OSR=32768

    //Set ADC interrupt
    DrvADC_ClearIntFlag();
    DrvADC_EnableInt();
    DrvADC_Enable();
    DrvADC_CombFilter(ENABLE);                 //Enable comb filter
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

13. Analog IP (CMP)

13.1. Example Name

HY16F188_CMP

13.2. Example Description

(1) CMP tutorial.
(2) Using #define to select to compile V12_Voltage_mueasure or RLO_Voltage_mueasure or CH3_Compare_CH2_Voltage.
(3) If select to compile V12_Voltage_mueasure, user can use meter to check CH1(PT1.0)-VSS pin to measure internal V12(REFO) voltage.
(4) If select to compileRLO_Voltage_mueasure, user can use meter to check CH1(PT1.0)-VSS pin. Observe Buit-in 16 nodes resistor to select different resistor nodes to output different voltages to the input channel RLO(VDD3V at this example) of the comparator
(5) If select to compile CH3_Compare_CH2_Voltage, compare CH3(PT1.2) with CH2(PT1.1) voltage. When CH3(PT1.2) is more than CH2(PT1.1), LCD dispaly ” 1” and CMPO output high. When CH2(PT1.1) is more than CH3(PT1.2), LCD dispaly ”0” and CMPO output low.

13.3. Program Description

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ------------------------------------------------------------------------
 * Project Name : HY16F188_CMP
 * IDE tooling   : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
```

```
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    ---------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA ---------------------
 *         GND    |
 *                |
 * ---------
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvCLOCK.h"
#include "DrvPMU.h"
#include "DrvREG32.h"
#include "HY2613.h"
#include "DrvI2C.h"
#include "DrvCMP.h"
#include "my define.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char    _byte;
   struct
```

```
  {
    unsigned b_ADCdone:1;
    unsigned b_TMAdone:1;
    unsigned b_TMBdone:1;
    unsigned b_TMC0done:1;
    unsigned b_TMC1done:1;
    unsigned b_CMPdone:1;
    unsigned b_UART_TxDone:1;
    unsigned b_UART_RxDone:1;
  };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define   I2CBufferSize   64

#define CMPO_PORT E_PT1
#define CMPOut BIT7

//#define V12_Voltage_mueasure
//#define RLO_Voltage_mueasure
#define CH3_Compare_CH2_Voltage
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS   MCUSTATUSbits;

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
```

```
void Delay(unsigned int num);
void InitaII2C(void);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

   InitaII2C();
   SYS_EnableGIE(4,0x3F);          //Enable GIE(Global Interrupt)
   MCUSTATUSbits._byte = 0;

   //Display Setting
   DisplayInit();
   ClearLCDframe();
   DisplayHYcon();

   //GPIO Setting & GPIO Setting
   DrvGPIO_Open(CMPO_PORT,CMPOut,E_IO_OUTPUT);   //Set PT1_7 OUTPUT

#if defined(RLO_Voltage_mueasure)
   DrvCMP_PInput(0);               //CMP positive input CH1
   DrvCMP_NInput(3);               //CMP negative input RLO
   DrvCMP_RLO_refV(2,1);           //RLO=VDD3V, CPIS=Closed
   DrvCMP_InputSwitch(1);          //INPUT SHORT SWITCH ENABLE
#endif

#if defined(V12_Voltage_mueasure)
   DrvPMU_VDDA_Voltage(E_VDDA2_4);
   DrvPMU_VDDA_LDO_Ctrl(E_LDO);
   DrvPMU_BandgapEnable();
   DrvPMU_REFO_Enable();
   DrvCMP_PInput(3);               //CMP positive input V12
   DrvCMP_NInput(0);               //CMP negative input CH1
   DrvCMP_InputSwitch(1);          //INPUT SHORT SWITCH ENABLE
#endif

#if defined(CH3_Compare_CH2_Voltage)
```

```
    unsigned int i=0;
    DrvCMP_PInput(2);                  //CMP positive input CH3
    DrvCMP_NInput(1);                  //CMP negative input CH2
    DrvCMP_InputSwitch(0);             //INPUT SHORT SWITCH ENABLE
#endif

    DrvCMP_Enable();                   //CMP enable.
    DrvCMP_OutputPinEnable(0);         //Enable CMP digital output to port
                                        //0 : PT1.7

#if defined(RLO_Voltage_mueasure)
    // Setting CPDM=0000b and CPDA to observe RLO voltage
    DrvCMP_RLO_Ctrl(0x00,0x00);    //CH1 to VSS around 0.0062V
    DrvCMP_RLO_Ctrl(0x01,0x00);    //CH1 to VSS around 0.2132V
    DrvCMP_RLO_Ctrl(0x02,0x00);    //CH1 to VSS around 0.4163V
    DrvCMP_RLO_Ctrl(0x03,0x00);    //CH1 to VSS around 0.6205V
    DrvCMP_RLO_Ctrl(0x04,0x00);    //CH1 to VSS around 0.8221V
    DrvCMP_RLO_Ctrl(0x05,0x00);    //CH1 to VSS around 1.0252V
    DrvCMP_RLO_Ctrl(0x06,0x00);    //CH1 to VSS around 1.2290V
    DrvCMP_RLO_Ctrl(0x07,0x00);    //CH1 to VSS around 1.4355V
    DrvCMP_RLO_Ctrl(0x08,0x00);    //CH1 to VSS around 1.6393V
    DrvCMP_RLO_Ctrl(0x09,0x00);    //CH1 to VSS around 1.8449V
    DrvCMP_RLO_Ctrl(0x0A,0x00);    //CH1 to VSS around 2.0476V
    DrvCMP_RLO_Ctrl(0x0B,0x00);    //CH1 to VSS around 2.2535V
    DrvCMP_RLO_Ctrl(0x0C,0x00);    //CH1 to VSS around 2.4577V
    DrvCMP_RLO_Ctrl(0x0D,0x00);    //CH1 to VSS around 2.6613V
    DrvCMP_RLO_Ctrl(0x0E,0x00);    //CH1 to VSS around 2.8686V
    DrvCMP_RLO_Ctrl(0x0F,0x00);    //CH1 to VSS around 3.0777V, VDD3V around 3.3091V
#endif

#if defined(CH3_Compare_CH2_Voltage)
    while(1)
    {
      i=DrvCMP_ReadData();
      LCD_DATA_DISPLAY(i);          //IF positive(CH3) > negative(CH2), CMPO=High, LCD Display(1). otherwise
CMPO=Low, LCD Display(0).
    }
#endif
```

```
    while(1);
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description    : I2C/UART/SPI interrupt Service Routine (HW0).            */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
    unsigned char I2C_Status,I2C_IntFlag;

    I2C_IntFlag=DrvI2C_ReadIntFlag();
    if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Interrupt Flag
    {
        I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
        switch(I2C_Status)
        {
            case 0x90:    //MACTFlag+RWFlag
                {          //START has been transmitted
                    DrvI2C_WriteData(I2C_TARGET);    //Send Slave Address & R/W Bit
                    DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
                    break;
                };
            case 0x84:    //MACTFlag+ACKFlag
                {          //Slave A + W has been transmitted. ACK has been received.
                    DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                    DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
                    break;
                };
            case 0x80: //MACTFlag
                {          //Slave A + W has been transmitted. ACK has been received.
                    DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                    DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
                    break;
                };
```

```
case 0x30:
    {
      DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
      I2C_EndFlag=1;
      break;
    };
case 0x8C:   //MACTFlag+DFFlag+ACKFlag
    {          //DATA has been transmitted and ACK has been received
      if(I2C_DataTxIndex<I2C_DataTxLen)
      {
        DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
        DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
      }
      else
      {
        if(I2C_RW == I2C_WRITE)
        {
          DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
          I2C_EndFlag=1;
        }
        else if(I2C_RW == I2C_READ)
          DrvI2C_Ctrl(1,0,0,0);   //I2C as master sends START signal
        I2C_DataTxIndex=0;
      }
      break;
    };
case 0x88:   //MACTFlag+DFFlag
    {          //DATA has been transmitted and NACK has been received
      DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
      I2C_DataTxIndex=0;
      I2C_EndFlag=1;
      break;
    };
case 0xB0:
    {          //A repeated START has been transmitted.
      DrvI2C_WriteData(I2C_TARGET | I2C_READ);   //Send Slave Address & R/W Bit
      DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
      break;
    }
```

```
    case 0x94:    //MACTFlag+RWFlag+ACKFlag
        {         //Slave A + R has been transmitted. ACK has been received.
          if(I2C_DataRxLen>1)
          {
            DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
          }else{
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          }
          break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {         //Data byte has been received. ACK has been transmitted.
          I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
          if((I2C_DataRxLen-1)>I2C_DataRxIndex)
          {
            DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
          }
          else
          {
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          }
          break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {         //Data byte has been received. NACK has been transmitted.
          I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
          DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
          I2C_EndFlag=1;
          break;
        };
    default:
        {
          DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          I2C_EndFlag=1;
          break;
        };
}
DrvI2C_ClearIRQ();
DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
```

```
    DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
  }
  if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
  {
    I2C_EndFlag=1;
    DrvI2C_ClearIRQ();
    DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
    DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
    DrvI2C_Ctrl(0,0,0,0);               //Clear all I2C flag
  }
  SYS_EnableGIE(4,0x3F);                //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description   : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description   : ADC interrupt Service Routine (HW2).                      */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
```

```
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{
  MCUSTATUSbits.b_CMPdone=1;
  DrvCMP_ClearIntFlag();          //Clear CMP interrupt flag
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description  : Exception Service Routines.                                */
```

```
/* Arguments       : None.                                                    */
/* Return Value : None.                                                    */
/* Remark          :                                                       */
/*-------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*-------------------------------------------------------------------------*/
/* Software Delay Subroutines                                              */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*-------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                          */
/* Description    : Hardware I2C Initial.                                  */
/* Arguments       : None.                                                 */
/* Return Value : None.                                                    */
/* Remark          :                                                       */
/*-------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}

/*-------------------------------------------------------------------------*/
/* End Of File                                                             */
/*-------------------------------------------------------------------------*/
```

14. Communication IP(SPI)

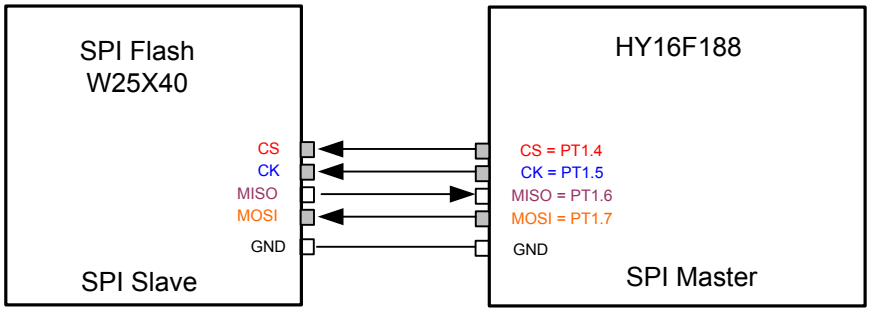
14.1. Example Name

HY16F188_SPI

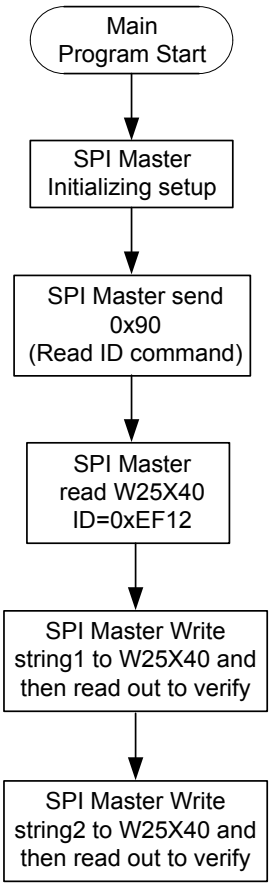
14.2. Example Description

(1) HY16F188 three-wire SPI Master mode tutorial.
(2)SPI Master communicate with SPI Flash 25X40(Winbond). Because this example code is three-wire SPI Master mode, so CS(Chip Select) funciton by using GPIO to do control, Set PT1.4=CS function.
(3) In this example code, first, read SPI Flash Manufacturer and Device Identification. If it is work correctly, variable Flash_ID is equal to 0xEF12
(4)Second, SPI Master writes a sequence of string "String1" to SPI flash and then read out to verify. If write and read data is equal, LCD Display "0", otherwise, LCD Display "1".
(5)Third, SPI Master writes a sequence of string "String2" to SPI flash and then read out to verify. If write and read data is equal, LCD Display "0", otherwise, LCD Display "123".

14.3. System Description



14.4. Software Flowchart



14.5. Program Description

Explanation : Paste main.c at here for reference only. No shows SPI Master initializing code below.

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------------
 * Project Name : HY16F188_SPI
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * HY16H188
 * ---------
 *              |                    W25Q16BV
 *              |                     --------
 *      PT1.4 | SPI_SS -----> |    CS    |
 *      PT1.5 | SPI_CLK ----> |    CLK |
 *      PT1.6 | SPI_MISO <--- |    DO    |
 *      PT1.7 | SPI_MOSI ---> |    DI    |
 *              |                     --------
 *              |                    ---------------------
```

```
 *      PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *      PT2.1 | SDA ---> SDA ---------------------
 *            |
 *            |
 * ---------
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvSPI32.h"
#include "DrvCLOCK.h"
#include "SPI_Flash.h"
#include "my define.h"
#include "HY2613.h"
#include "DrvI2C.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;
```

```
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define   I2CBufferSize   64
#define   FlashAddress 0x000000

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS   MCUSTATUSbits;
unsigned char Flash_Write_Buffer[256];
unsigned char Flash_Read_Buffer[256];
unsigned short Flash_ID;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

  const unsigned char String1[]={
      "Hycon Technology was established in July, 2007 in Taipei, Taiwan. We dedicate ourselves to develop high precision
and low drift analog signal related processing ICs." };
  const unsigned char String2[]={
      "The identity stands for the vision and values of Hycon Technology- To be the ideal solution partner in the area of
analog circuit." };
  unsigned short index_d,Size;
```

```
InitalSPI();
InitalI2C();

SYS_EnableGIE(4,0x3F);                                  // Enable GIE(Global Interrupt)

DisplayInit();
ClearLCDframe();
Delay(10000);
DisplayHYcon();
Delay(10000);
MCUSTATUSbits._byte = 0;

Flash_ID=SpiFlash_ReadMidDid();                         //If correct, Flash_ID=0xEF12

//Test1 : 16F write String1 to Flash, and Read out
Size=sizeof(String1);
SpiFlash_ChipErase();                                   //Erase DATA
Delay(256);
SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);         //Read DATA
Delay(256);

for( index_d=0;index_d<Size;index_d++)
{
   Flash_Write_Buffer[index_d]=String1[index_d];
}
SpiFlash_PageProgram(Flash_Write_Buffer,FlashAddress,Size);    //Write DATA
Delay(256);
SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);         //Read DATA
Delay(256);

for( index_d=0;index_d<Size;index_d++)
{
   if(Flash_Read_Buffer[index_d]!=String1[index_d])
   {
      LCD_DATA_DISPLAY(1);
      while(1);
   }
   else
```

```
        LCD_DATA_DISPLAY(0);
    }

  //Test2 : 16F write String2 to Flash, and Read out
  Size=sizeof(String2);
  SpiFlash_SectorErase(0x00);                                    //Erase DATA
  Delay(256);
  SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);        //Read DATA
  Delay(256);

  for( index_d=0;index_d<Size;index_d++)
  {
    Flash_Write_Buffer[index_d]=String2[index_d];
  }
  SpiFlash_PageProgram(Flash_Write_Buffer,FlashAddress,Size);   //Write DATA
  Delay(256);
  SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);        //Read DATA
  Delay(256);

  for( index_d=0;index_d<Size;index_d++)
  {
    if(Flash_Read_Buffer[index_d]!=String2[index_d])
    {
      LCD_DATA_DISPLAY(1);                                       //If error, LCD Display "1"
      while(1);
    }
  else
      LCD_DATA_DISPLAY(123);                                     //If correct, LCD Display "123"
    }

  while(1);
  return 0;

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments      : None.                                                    */
```

```
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW0_ISR(void)
{
   unsigned char I2C_Status,I2C_IntFlag;

   I2C_IntFlag=DrvI2C_ReadIntFlag();
   if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
   {
      I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
      switch(I2C_Status)
      {
         case 0x90:   //MACTFlag+RWFlag
              {          //START has been transmitted
                 DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
                 DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
                 break;
              };
         case 0x84:   //MACTFlag+ACKFlag
              {          //Slave A + W has been transmitted. ACK has been received.
                 DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                 DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                 break;
              };
         case 0x80: //MACTFlag
              {         //Slave A + W has been transmitted. ACK has been received.
                 DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                 DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                 break;
              };
         case 0x30:
              {
                 DrvI2C_Ctrl(0,0,0,0);         //Clear all I2C flag
                 I2C_EndFlag=1;
                 break;
              };
         case 0x8C:   //MACTFlag+DFFlag+ACKFlag
              {          //DATA has been transmitted and ACK has been received
```

```
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
               DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
               DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
            }
            else
            {
               if(I2C_RW == I2C_WRITE)
               {
                  DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                  I2C_EndFlag=1;
               }
               else if(I2C_RW == I2C_READ)
                  DrvI2C_Ctrl(1,0,0,0);    //I2C as master sends START signal
               I2C_DataTxIndex=0;
            }
            break;
         };
      case 0x88:    //MACTFlag+DFFlag
         {          //DATA has been transmitted and NACK has been received
            DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
            I2C_EndFlag=1;
            break;
         };
      case 0xB0:
         {          //A repeated START has been transmitted.
            DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;
         }
      case 0x94:    //MACTFlag+RWFlag+ACKFlag
         {          //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
               DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }else{
               DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
```

```
            break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {         //Data byte has been received. ACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            if((I2C_DataRxLen-1)>I2C_DataRxIndex)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }
            else
            {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {         //Data byte has been received. NACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
            I2C_EndFlag=1;
            break;
        };
    default:
        {
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            I2C_EndFlag=1;
            break;
        };
  }
  DrvI2C_ClearIRQ();
  DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
  DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
}
if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
{
  I2C_EndFlag=1;
  DrvI2C_ClearIRQ();
  DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
  DrvI2C_ClearIntFlag(1);             //Clear I2C Interrupt Flag(I2CEIF)
```

```
    DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
  }
  SYS_EnableGIE(4,0x3F);               //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description  : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).   */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description  : ADC interrupt Service Routine (HW2).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{
```

```
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}
```

```
/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
/* Description    : Hardware I2C Initial.                                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalI2C(void)
{
   DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
   DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

15. Communication IP(UART)

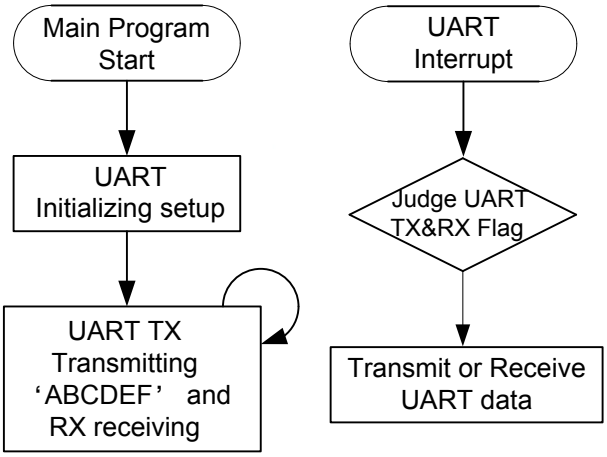
15.1. Example Name

HY16F188_UART

15.2. Example Description

(1) HY16F188 UART TX and RX tutorial.
(2) UART Port2, TX=PT2.2, RX=PT2.3
(3) using #define HAO_2MHZ/ HAO_4MHZ/ HAO_10MHZ/ HAO_16MHZ to select HAO frequency.
(4) UART TX continue transmit 'ABCEDF' until RX receiving 'abcedf' to stop transmit. If RX receiving not equal to 'abcedf', UART TX start to transmit 'ABCEDF'

15.3. Software Flowchart



15.4. Program Description

```
/*****************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------
 * Project Name : HY16F188_UART
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                    --------------------
 *       PT2.0 | SCL ---> SCL | LCD Drive HY2613 |
 *       PT2.1 | SDA ---> SDA --------------------
 *       PT2.2 | TX    --->
 *       PT2.3 | RX    <---
 *         GND    |
 *                |
 * ---------
```

```
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvUART.h"
#include "DrvI2C.h"
#include "DrvCLOCK.h"
#include "HY2613.h"
#include "my define.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_UART_TxDone:1;
      unsigned b_UART_RxDone:1;
   };
} MCUSTATUS;


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define   I2CBufferSize   64
#define I2C_WRITE          0x00          // I2C WRITE command
#define I2C_READ           0x01          // I2C READ command
```

```
#define HSRC    //Internal HAO
//#define HSXT    //External 4MHz

#define HAO_2MHZ
//#define HAO_4MHZ
//#define HAO_10MHZ

#define    UART_PORT E_PT2
#define    UART_TXD    BIT2
#define    UART_RXD    BIT3
#define    UartBufferSize 128
#define    Uart_RX_BufferSize 6
#define    Uart_TX_BufferSize 8
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_RW;
unsigned char I2C_TARGET;
unsigned char I2C_EndFlag;
unsigned int I2C_Sendbuf[I2CBufferSize];
unsigned int I2C_Recbuf[I2CBufferSize];
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;

MCUSTATUS    MCUSTATUSbits;

unsigned char UartRxBuffer[Uart_RX_BufferSize]={0},UartTxBuffer[Uart_TX_BufferSize]={0};
unsigned char UartTxIndex,UartTxLength,UartRxIndex,UartRxLength;
unsigned char UartRxBuffer_Command[Uart_RX_BufferSize]=
{
0x61,0x62,0x63,0x64,0x65,0x66    //ASCII    KEYWORD = abcdef
};
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalI2C(void);
void InitalUart(void);
void InitalClock(void);
```

```
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

  unsigned char Stop_To_Send_UART=DISABLE;
  InitalClock();
  InitalI2C();
  InitalUart();

  MCUSTATUSbits._byte = 0;
  MCUSTATUSbits.b_UART_TxDone=ENABLE;
  SYS_EnableGIE(4,0x3F);    //Enable GIE(Global Interrupt)

  DisplayInit();
  ClearLCDframe();
  Delay(10000);
  DisplayHYcon();
  Delay(50000);

  UartTxIndex=0;
  UartRxIndex=0;

  while(1)
  {

    //UART RX
    if(MCUSTATUSbits.b_UART_RxDone==ENABLE)
    {

      if(
        (UartRxBuffer[5]==UartRxBuffer_Command[5] && UartRxBuffer[4]==UartRxBuffer_Command[4]) &&
        (UartRxBuffer[3]==UartRxBuffer_Command[3] && UartRxBuffer[2]==UartRxBuffer_Command[2]) &&
        (UartRxBuffer[1]==UartRxBuffer_Command[1] && UartRxBuffer[0]==UartRxBuffer_Command[0])
        )
    {
        //if UART receive == 0x61(a)0x62(b)0x63(c)0x64(d)0x65(e)0x66(f)
        //send out 0x61(a)0x62(b)0x63(c)0x64(d)0x65(e)0x66(f) and stop to send out
```

```
            Stop_To_Send_UART=ENABLE;    //UART stop to send out ABCDEF
            for(UartTxLength=0;UartTxLength<=Uart_TX_BufferSize;UartTxLength++)
            {
              UartTxBuffer[UartTxLength]=UartRxBuffer[UartTxLength];
              if(UartTxLength==Uart_RX_BufferSize)
              {
                UartRxIndex=0;
                MCUSTATUSbits.b_UART_TxDone=DISABLE;
                DrvUART_EnableInt(ENABLE,DISABLE);    //Enable UART Tx Interrupt, Disable UART Rx Interrup
                while(!MCUSTATUSbits.b_UART_TxDone);    //If    MCUSTATUSbits.b_UART_TxDone=DISABLE, stop at
here
              }
            }
        }
        else
        {
          //if UART receive != 0x61(a)0x62(b)0x63(c)0x64(d)0x65(e)0x66(f)
          //User defined at here
          Stop_To_Send_UART=DISABLE;    //UART start to send out ABCDEF
        }
          UartRxIndex=0;    //When finished the data reception. Set UartRxIndex=0
          MCUSTATUSbits.b_UART_RxDone=DISABLE;
        }

        //UART TX
        if(MCUSTATUSbits.b_UART_TxDone==ENABLE && Stop_To_Send_UART==DISABLE )
        {
          UartTxBuffer[7]='\r';
          UartTxBuffer[6]='\n';
          UartTxBuffer[5]=0x46;    //F
          UartTxBuffer[4]=0x45;    //E
          UartTxBuffer[3]=0x44;    //D
          UartTxBuffer[2]=0x43;    //C
          UartTxBuffer[1]=0x42;    //B
          UartTxBuffer[0]=0x41;    //A
          MCUSTATUSbits.b_UART_TxDone=DISABLE;
          UartTxLength=8;
          UartTxIndex=0;
          DrvUART_EnableInt(ENABLE,ENABLE);    //Enable UART Tx Interrupt;Enable UART Rx Interrupt
```

```
        while(!MCUSTATUSbits.b_UART_TxDone);    //If    MCUSTATUSbits.b_UART_TxDone=DISABLE, stop at here
      }

   }


}
/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description    : I2C/UART/SPI interrupt Service Routine (HW0).            */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
   unsigned char I2C_Status,I2C_IntFlag;

   if(DrvUART_GetRxFlag())
   {
      UartRxBuffer[UartRxIndex]=DrvUART_Read();
      UartRxIndex++;
      if(UartRxIndex>=Uart_RX_BufferSize)
      {
         UartRxIndex=0;
         MCUSTATUSbits.b_UART_RxDone=ENABLE;
      }
         DrvUART_ClrRxFlag();
   }

   if(DrvUART_GetTxFlag())
   {
      if(MCUSTATUSbits.b_UART_TxDone==DISABLE)
      {
         DrvUART_Write(UartTxBuffer[UartTxIndex++]);
         DrvUART_ClrTxFlag();
         if(UartTxIndex>=UartTxLength)
         {
            DrvUART_EnableInt(ENABLE,ENABLE);    //ENABLE UART Tx Interrupt, ENABLE UART Rx Interrupt
```

```
            MCUSTATUSbits.b_UART_TxDone=ENABLE;
            UartTxIndex=0;
        }
    }
    if(MCUSTATUSbits.b_UART_TxDone==ENABLE)
    {
        DrvUART_EnableInt(DISABLE,ENABLE);    //DISABLE UART Tx Interrupt, ENABLE UART Rx Interrupt
        DrvUART_ClrTxFlag();
    }
}

I2C_IntFlag=DrvI2C_ReadIntFlag();
if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Interrupt Flag
{
    I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
    switch(I2C_Status)
    {
        case 0x90:    //MACTFlag+RWFlag
            {          //START has been transmitted
                DrvI2C_WriteData(I2C_TARGET);    //Send Slave Address & R/W Bit
                DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
                break;
            };
        case 0x84:    //MACTFlag+ACKFlag
            {          //Slave A + W has been transmitted. ACK has been received.
                DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
                break;
            };
        case 0x80: //MACTFlag
            {         //Slave A + W has been transmitted. ACK has been received.
                DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
                break;
            };
        case 0x30:
            {
                DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
                I2C_EndFlag=1;
```

```
            break;
        };
    case 0x8C:    //MACTFlag+DFFlag+ACKFlag
        {           //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
                DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);    //Send Data to Slave
                DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
            }
            else
            {
                if(I2C_RW == I2C_WRITE)
                {
                    DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                    I2C_EndFlag=1;
                }
                else if(I2C_RW == I2C_READ)
                    DrvI2C_Ctrl(1,0,0,0);    //I2C as master sends START signal
                I2C_DataTxIndex=0;
            }
            break;
        };
    case 0x88:    //MACTFlag+DFFlag
        {           //DATA has been transmitted and NACK has been received
            DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
            I2C_EndFlag=1;
            break;
        };
    case 0xB0:
        {           //A repeated START has been transmitted.
            DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;
        }
    case 0x94:    //MACTFlag+RWFlag+ACKFlag
        {           //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
```

```
            DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
          }else{
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          }
          break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {           //Data byte has been received. ACK has been transmitted.
          I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
          if((I2C_DataRxLen-1)>I2C_DataRxIndex)
          {
            DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
          }
          else
          {
            DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          }
          break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {           //Data byte has been received. NACK has been transmitted.
          I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
          DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
          I2C_EndFlag=1;
          break;
        };
    default:
        {
          DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
          I2C_EndFlag=1;
          break;
        };
  }
  DrvI2C_ClearIRQ();
  DrvI2C_ClearEIRQ();                 //Clear EIRQFlag
  DrvI2C_ClearIntFlag(0);             //Clear I2C Interrupt Flag(I2CIF)
}
if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
{
```

```
        I2C_EndFlag=1;
        DrvI2C_ClearIRQ();
        DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
        DrvI2C_ClearIntFlag(1);              //Clear I2C Interrupt Flag(I2CEIF)
        DrvI2C_Ctrl(0,0,0,0);                //Clear all I2C flag
    }
    SYS_EnableGIE(4,0x3F);                   //Enable GIE(Global Interrupt)
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description   : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description   : ADC interrupt Service Routine (HW2).                      */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description   : CMP & OPA interrupt Service Routine (HW3).                */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
```

```
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description  : PT2 interrupt Service Routine (HW5).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
```

```
    asm("nop");    //procedure define by customer.
    asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for(;num>0;num--)
    asm("NOP");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalI2C(void)                                            */
/* Description    : Hardware I2C Initial.                                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalI2C(void)
{

    DrvI2C_SetIOPin(4);       //setting io pin, 4 SCL=PT2.0;SDA=PT2.1
#if defined(HSRC)
    #if defined(HAO_2MHZ)
    DrvI2C_Open(0x4);          //Enable I2C function and set I2C baud rate=100kHz
    //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
    #endif
    #if defined(HAO_4MHZ)
    DrvI2C_Open(0x9);          //Enable I2C function and set I2C baud rate=100kHz
    //Default CPU clock is 4MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 4000000/[4*(9+1)]=100kHz
    #endif
    #if defined(HAO_10MHZ)
    DrvI2C_Open(0x18);          //Enable I2C function and set I2C baud rate=100kHz
    //Default CPU clock is 10MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 10000000/[4*(24+1)]=100kHz
    #endif
#endif
#if defined(HSXT)
```

```
    DrvI2C_Open(0x9);          //Enable I2C function and set I2C baud rate=100kHz
    //Default CPU clock is 4MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 4000000/[4*(9+1)]=100kHz
#endif

    DrvI2C_EnableInt(2);     //Enable I2C interrupt and error interrupt
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalClock()                                              */
/* Description    : CLOCK Initial Subroutines.                               */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalClock(void)
{

#if defined(HSRC)
    #if defined(HAO_2MHZ)
    //Clock INIT 2MHZ
    DrvCLOCK_EnableHighOSC(E_INTERNAL,10);          //Select HSRC
    DrvCLOCK_SelectIHOSC(0);                        //Select internal 2MHZ
    DrvCLOCK_SelectMCUClock(0,0);                   //CPU CLOCK IS 'hs_ck/1'
    DrvCLOCK_CalibrateHAO(0);                             //Calibration 2.000MHz
    #endif
    #if defined(HAO_4MHZ)
    //Clock INIT 4MHZ
    DrvCLOCK_EnableHighOSC(E_INTERNAL,10);          //Select HSRC
    DrvCLOCK_SelectIHOSC(1);                        //Select internal 4MHZ
    DrvCLOCK_SelectMCUClock(0,0);                   //CPU CLOCK IS 'hs_ck/1'
    DrvCLOCK_CalibrateHAO(1);                             //Calibration 4.000MHz
    #endif
    #if defined(HAO_10MHZ)
    //Clock INIT 10MHZ
    DrvCLOCK_EnableHighOSC(E_INTERNAL,10);          //Select HSRC
    DrvCLOCK_SelectIHOSC(2);                        //Select internal 10MHZ
    DrvCLOCK_SelectMCUClock(0,0);                   //CPU CLOCK IS 'hs_ck/1'
    DrvCLOCK_CalibrateHAO(2);                             //Calibration 10.000MHz
    #endif
```

```
#endif

#if defined(HSXT)
   DrvCLOCK_EnableHighOSC (E_EXTERNAL,50);
#endif
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalUart()                                               */
/* Description    : UART Initial Subroutines.                                */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalUart(void)
{

#if defined(HSRC)
   DrvUART_ClkEnable(1,0);                  //Choose the internal HAO as clock source
   #if defined(HAO_2MHZ)
   DrvUART_Open(2000,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,5);
   #endif
   #if defined(HAO_4MHZ)
   DrvUART_Open(4000,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,5);
   #endif
   #if defined(HAO_10MHZ)
   DrvUART_Open(10000,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,5);
   #endif
#endif

#if defined(HSXT)
   DrvUART_ClkEnable(0,0);                  //Choose the external OSC as clock source
   DrvUART_Open(4000,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,5);
#endif

   //2000 : oscillator frequency 2MHz Unit After Calibration HAO = 2000kHz
   //4000 : oscillator frequency 4MHz Unit After Calibration HAO = 4000kHz
   //10000 : oscillator frequency 10MHz Unit After Calibration HAO = 10000kHz
   //None parity
```

```
    //8 data bits.
    //5 : Port 2.2 =TX, Port 2.3 =RX

    DrvGPIO_Open(UART_PORT,UART_TXD,E_IO_OUTPUT);
    DrvGPIO_Open(UART_PORT,UART_RXD,E_IO_INPUT);
    DrvGPIO_Open(UART_PORT,UART_RXD|UART_TXD,E_IO_PullHigh);

    DrvGPIO_ClkGenerator(E_HS_CK,1);
    DrvUART_EnableInt(ENABLE,ENABLE);    //Enable UART Tx Interrupt, Enable UART Rx Interrupt
    DrvUART_Close();
    DrvUART_Enable();
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

16. Communication IP(I2C)

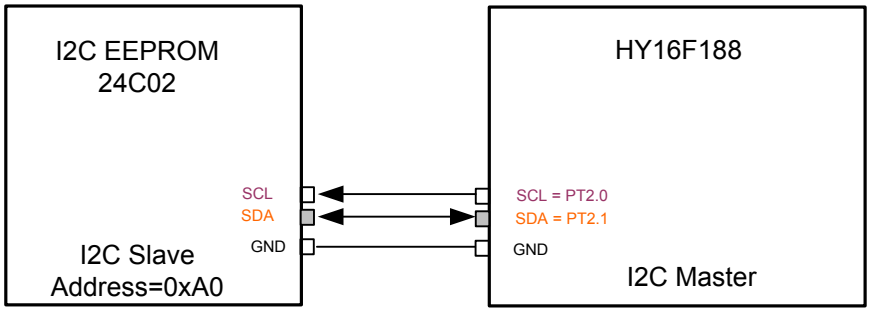
16.1. Example Name

HY16F188_I2C

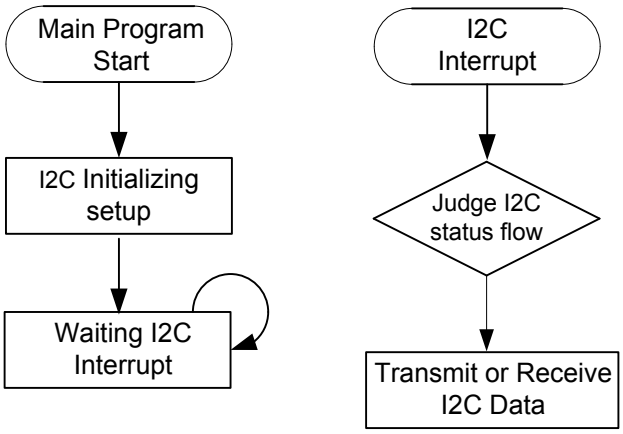
16.2. Example Description

(1) HY16F188 I2C Master mode tutorial.
(2)I2C Master communicate with I2C EEPROM 24C02, I2C Master single write & read and multiple write & read example.
(3) In this example, first, I2C Master write 0x01 to EEPROM WORD ADDRESS 0x00, and then I2C Master read EEPROM WORD ADDRESS 0x00.
(4) Second, I2C Master write a sequence of 4 bytes data to EEPROM WORD ADDRESS 0x01, and then I2C Master read a sequence of 5 bytes data from EEPROM WORD ADDRESS 0x00.
(5) If finish this example correctly, EEPROM address 0x00 data is equal to 0x01, EEPROM address 0x01 data is equal to 0x02, EEPROM address 0x02 data is equal to 0x03, EEPROM address 0x03 data is equal to 0x04, EEPROM address 0x04 data is equal to 0x05.

16.3. System Description



16.4. Software Flowchart



16.5. Program Description

Explanation : Paste main.c at here for reference only. No shows I2C Master initializing code below.

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ------------------------------------------------------------------------
 * Project Name : HY16F188_I2C
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.    : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device     :
```

```
 * Description    :
 * Created Date : 2018/2/18
 * Created by     :
 *
 * Program Description:
 * --------------------
 *
 * ---------
 * HY16F188 |                        ---------------------
 *       PT2.0 | SCL ---> SCL | EEPROM 24LC02       |
 *       PT2.1 | SDA ---> SDA ---------------------
 *         GND    |
 *                |
 * ---------
 *******************************************************************************/
/*------------------------------------------------------------------------------*/
/* Includes                                                                     */
/*------------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "DrvI2C.h"
#include "System.h"
#include "EEPROM_24Cxx.h"
#include "my define.h"

/*------------------------------------------------------------------------------*/
/* STRUCTURES                                                                   */
/*------------------------------------------------------------------------------*/



/*------------------------------------------------------------------------------*/
/* DEFINITIONS                                                                  */
/*------------------------------------------------------------------------------*/
#define    I2CBufferSize    256



/*------------------------------------------------------------------------------*/
/* Global CONSTANTS                                                             */
/*------------------------------------------------------------------------------*/
unsigned char I2C_Read_Buffer[I2CBufferSize];;
```

```
unsigned char I2C_RW;
unsigned char I2C_TARGET;                    //Target I2C slave address
unsigned char I2C_Sendbuf[I2CBufferSize];
unsigned char I2C_Recbuf[I2CBufferSize];
unsigned char I2C_EndFlag;
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;


/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
   unsigned int i;
   unsigned char EEPROM_WriteData[4] = {0x02,0x03,0x04,0x05};

   for(i=0;i<=I2CBufferSize;i++)
    {
       I2C_Read_Buffer[i]=0;                 //Initial I2C data buffer=0
    }

   DrvI2C_SetIOPin(4);                       //Setting io pin, 4: SCL=PT2.0;SDA=PT2.1
   DrvI2C_Open(0x63);                        //Enable I2C function and set I2C baud rate=5kHz
   //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(99+1)]=5kHz
   DrvI2C_EnableInt(2);                      //Enable I2C interrupt and error interrupt

   SYS_EnableGIE(4,0x3F);                    //Enable GIE(Global Interrupt)

   //I2C single I2C_WRITE & I2C_READ
   EEPROM_ByteWrite(0x00,0x01);              //Single Byte I2C_WRITE word address 0x00, and I2C_WRITE
data 0x01
   Delay(1000);                              //Delay for EEPROM 24C02 I2C_WRITE Cycle Time 5ms
   I2C_Read_Buffer[0]=EEPROM_ByteRead(0x00);      //Single Byte I2C_READ word address 0x00, the I2C_READ
value is 0x01
```

```
    Delay(1000);                                    //Delay for EEPROM 24C02 I2C_WRITE Cycle Time 5ms

    // I2C sequential I2C_WRITE & I2C_READ
    EEPROM_WriteArray(0x01,EEPROM_WriteData,4);   //I2C_WRITE array data to the word address from 0x01 to 0x04
    Delay(1000);                                    //Delay for EEPROM 24C02 I2C_WRITE Cycle Time 5ms
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 5);   //sequential I2C_READ data from 0x00 to 0x04
    Delay(1000);                                    //Delay for EEPROM 24C02 I2C_WRITE Cycle Time 5ms

    while(1);

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
    unsigned char I2C_Status,I2C_IntFlag;

    I2C_IntFlag=DrvI2C_ReadIntFlag();
    if((I2C_IntFlag == E_DRVI2C_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))   //Get I2C Interrupt Flag
    {
      I2C_Status=DrvI2C_GetStatusFlag();        //Get I2C Status Flag
      switch(I2C_Status)
      {
        case 0x90:   //MACTFlag+RWFlag
            {          //START has been transmitted
              DrvI2C_WriteData(I2C_TARGET);   //Send Slave Address & R/W Bit
              DrvI2C_Ctrl(0,0,0,0);              //Clear all I2C flag
              break;
            };
        case 0x84:   //MACTFlag+ACKFlag
            {          //Slave A + W has been transmitted. ACK has been received.
              DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
              DrvI2C_Ctrl(0,0,0,0);          //Clear all I2C flag
```

```
            break;
        };
    case 0x80: //MACTFlag
        {       //Slave A + W has been transmitted. ACK has been received.
            DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;
        };
    case 0x30:
        {
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
            I2C_EndFlag=1;
            break;
        };
    case 0x8C:  //MACTFlag+DFFlag+ACKFlag
        {       //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
                DrvI2C_WriteData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
                DrvI2C_Ctrl(0,0,0,0);       // Clear all I2C flag
            }
            else
            {
                if(I2C_RW == I2C_WRITE)
                {
                    DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
                    I2C_EndFlag=1;
                }
                else if(I2C_RW == I2C_READ)
                    DrvI2C_Ctrl(1,0,0,0);   //I2C as master sends START signal
                I2C_DataTxIndex=0;
            }
            break;
        };
    case 0x88:  //MACTFlag+DFFlag
        {       //DATA has been transmitted and NACK has been received
            DrvI2C_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
            I2C_EndFlag=1;
```

```
            break;
        };
    case 0xB0:
        {           //A repeated START has been transmitted.
            DrvI2C_WriteData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;
        }
    case 0x94:    //MACTFlag+RWFlag+ACKFlag
        {           //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }else{
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
        {           //Data byte has been received. ACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            if((I2C_DataRxLen-1)>I2C_DataRxIndex)
            {
                DrvI2C_Ctrl(0,0,0,1);    //Set ACK bit
            }
            else
            {
                DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
        };
    case 0x98:    //MACTFlag+RWFlag+DFFlag
        {           //Data byte has been received. NACK has been transmitted.
            I2C_Recbuf[I2C_DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
            I2C_EndFlag=1;
            break;
        };
    default:
```

```
            {
               DrvI2C_Ctrl(0,0,0,0);    //Clear all I2C flag
               I2C_EndFlag=1;
               break;
            };
      }
      DrvI2C_ClearIRQ();
      DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
      DrvI2C_ClearIntFlag(0);              //Clear I2C Interrupt Flag(I2CIF)
   }
   if((I2C_IntFlag == E_DRVI2C_ERROR_INT)||(I2C_IntFlag == E_DRVI2C_INT_ALL))    //Get I2C Error Interrupt Flag
   {
      I2C_EndFlag=1;
      DrvI2C_ClearIRQ();
      DrvI2C_ClearEIRQ();                  //Clear EIRQFlag
      DrvI2C_ClearIntFlag(1);              //Clear I2C Interrupt Flag(I2CEIF)
      DrvI2C_Ctrl(0,0,0,0);                //Clear all I2C flag
   }
   SYS_EnableGIE(4,0x3F);                  //Enable GIE(Global Interrupt)

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description    : ADC interrupt Service Routine (HW2).                     */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
```

```
/* Remark          :                                                    */
/*---------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                            */
/* Description   : CMP & OPA interrupt Service Routine (HW3).          */
/* Arguments     : None.                                               */
/* Return Value : None.                                                */
/* Remark          :                                                   */
/*---------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                            */
/* Description   : PT1 interrupt Service Routine (HW4).                */
/* Arguments     : None.                                               */
/* Return Value : None.                                                */
/* Remark          :                                                   */
/*---------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*---------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                            */
/* Description   : PT2 interrupt Service Routine (HW5).                */
/* Arguments     : None.                                               */
/* Return Value : None.                                                */
/* Remark          :                                                   */
/*---------------------------------------------------------------------*/
void HW5_ISR(void)
```

```
{


}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}



/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

17. Peripheral IP(Power)

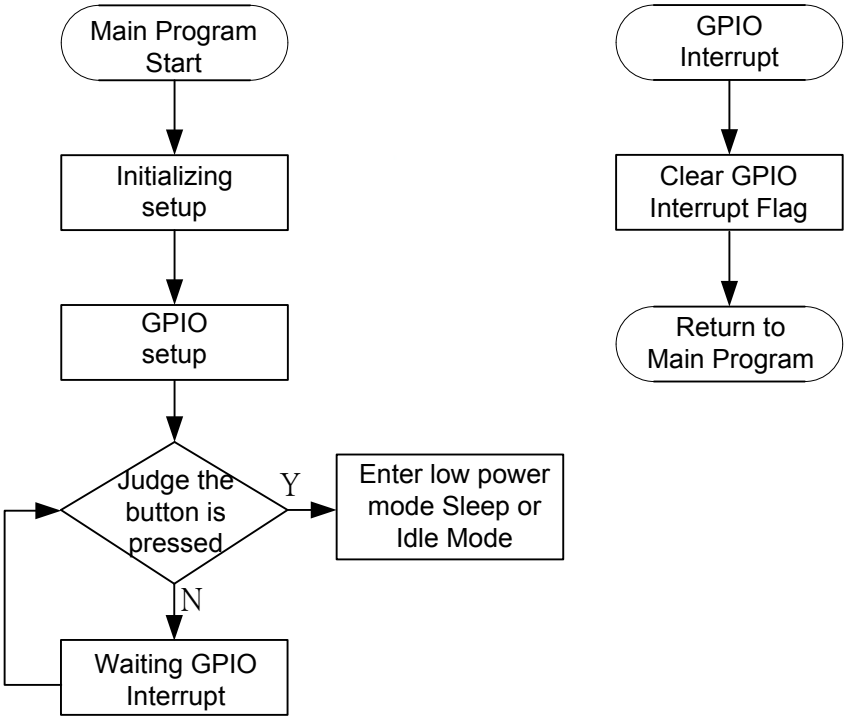
17.1. Example Name

HY16F188_Power

17.2. Example Description

(1) HY16F188 Sleep and Idle mode tutorial
(2) If press button PT1.6, enter Sleep mode.
(3) If press button PT1.7, enter Idle mode.
(4) If this example code work on HY16F188 Development board, user can measure Sleep mode current about 2.5uA, Idle mode current about 3.5uA.

17.3. Software Flowchart



17.4. Program Description

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ------------------------------------------------------------------------
 * Project Name : HY16F188_Power
 * IDE tooling   : AndeSight C/C++ IDE, version: 2.1.1 Build ID : 201608241332
 * Device Ver.   : HY16F_RDSp3_DeviceV0.2 crt0.o for HY16F18x MCU.
 * Library Ver. : 1.5
 * MCU Device    :
 * Description   :
 * Created Date : 2018/2/18
 * Created by    :
 *
 * Program Description:
 * --------------------
 *
 *******************************************************************************/
/*------------------------------------------------------------------------------*/
/* Includes                                                                     */
/*------------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvCLOCK.h"
#include "DrvPMU.h"
#include "my define.h"
#include "DrvGPIO.h"

/*------------------------------------------------------------------------------*/
/* STRUCTURES                                                                   */
/*------------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_ADCdone:1;
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMC0done:1;
```

```
        unsigned b_WDTdone:1;
        unsigned b_RTCdone:1;
        unsigned b_UART_TxDone:1;
        unsigned b_UART_RxDone:1;
    };
} MCUSTATUS;

typedef union _PTINTSTATUS
{
    char    _byte;
    struct
    {
        unsigned b_PTINT0done:1;
        unsigned b_PTINT1done:1;
        unsigned b_PTINT2done:1;
        unsigned b_PTINT3done:1;
        unsigned b_PTINT4done:1;
        unsigned b_PTINT5done:1;
        unsigned b_PTINT6Done:1;
        unsigned b_PTINT7Done:1;
    };
} PTINTSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#define KEY_PORT E_PT1
#define KEYIN0 BIT6
#define KEYIN1 BIT7
#define KEYIN0_PIN 6
#define KEYIN1_PIN 7


/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS    MCUSTATUSbits;
PTINTSTATUS    PT1INTSTATUSbits;

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{

    DrvGPIO_ClkGenerator(E_HS_CK,1);                    //Set IO sampling clock input source is HS_CK
    DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_INPUT);        //set PT1.7/PT1.6 INPUT
```

```
    DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_PullHigh);     //enable PT1.7/PT1.6 pull high R
    DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_IntEnable);   //PT1.7/PT1.6 interrupt enable
    DrvGPIO_IntTrigger(KEY_PORT,KEYIN1|KEYIN0,E_N_Edge);   //PT1.7/PT1.6 interrupt trigger method is negative
edge
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);              //clear PT1 interrupt flag
    PT1INTSTATUSbits._byte = 0;
    SYS_EnableGIE(4,0x3F);                                     //Enable GIE(Global Interrupt)

    while(1)
    {
      if(PT1INTSTATUSbits.b_PTINT6Done)                        //if PT1.6 low
      {
        PT1INTSTATUSbits.b_PTINT6Done=0;
        //If wake up from sleep mode, user can enable internal HAO first to do some application
        DrvPMU_LDO_LowPower(0);                                //SET normal power mode
        DrvCLOCK_EnableHighOSC(E_INTERNAL,1);
        DrvCLOCK_SelectMCUClock(E_HS_CK,0);                    //SET CPUCKL=HAO/1

        //Enter sleep mode setting
        DrvPMU_LDO_LowPower(1);                                //SET low power mode
        DrvCLOCK_SelectMCUClock(E_LS_CK,1);                    //SET CPUCKL=LPO/2
        DrvCLOCK_CloseIHOSC();                                 //Close HAO
        SYS_LowPower(0);                                       //sleep mode, around 2.5uA

      }

      if(PT1INTSTATUSbits.b_PTINT7Done)                        //if PT1.7 low
      {
        PT1INTSTATUSbits.b_PTINT7Done=0;
        //If wake up from idle mode, user can enable internal HAO first to do some application
        DrvPMU_LDO_LowPower(0);                                //SET normal power mode
        DrvCLOCK_EnableHighOSC(E_INTERNAL,1);
        DrvCLOCK_SelectMCUClock(E_HS_CK,0);                    //SET CPUCKL=HAO/1

        //Enter idle mode setting
        DrvPMU_LDO_LowPower(1);                                //SET low power mode
        DrvCLOCK_SelectMCUClock(E_LS_CK,1);                    //SET CPUCKL=LPO/2
        DrvCLOCK_CloseIHOSC();                                 //Close HAO
        SYS_LowPower(1);                                       //Idle Mode, around 3.5uA

      }
    }
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
```

```
void HW0_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description  : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).   */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description  : ADC interrupt Service Routine (HW2).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description  : CMP & OPA interrupt Service Routine (HW3).                 */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description  : PT1 interrupt Service Routine (HW4).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{
  uint32_t PORT_IntFlag;
```

```
    PORT_IntFlag=DrvGPIO_GetIntFlag(KEY_PORT);
    if((PORT_IntFlag&KEYIN0)==KEYIN0)
    {
      PT1INTSTATUSbits.b_PTINT6Done=1;
    }
    if((PORT_IntFlag&KEYIN1)==KEYIN1)
    {
      PT1INTSTATUSbits.b_PTINT7Done=1;
    }
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);          //clear PT1_7/1_6 interrupt flag
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description   : PT2 interrupt Service Routine (HW5).                      */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Function Name: tlb_exception_handler()                                    */
/* Description   : Exception Service Routines.                               */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
    asm("nop"); //procedure define by customer.
    asm("nop");
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for(;num>0;num--)
    asm("NOP");
}


/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

18. Revisions

The following describes the major changes made to the document, excluding the punctuation and font changes.

Version	Date	Page	Summary of Changes
V03	2013/12/1	All	First Edition
V06	2017/1/17	All	Modify all document content and IP demo code.
V07	2018/6/15	All	- Change the single bin file to separate 3 bin files(APP Bin File/Data Bin File/Bin File) - All : Modify the I2C_EndFlag judgement - HY16F188_ADC : Add HAO 2MHz/4MHz/10MHz/16MHz clock setting. - HY16F188_GPIO : Modify the GPIO initialization(GPIO setting-->Clear GPIO Flag-->judgement GPIO INT) - HY16F188_UART : Modify the UART initialization, avoid to occur the first byte error issue. Modify the demo code application. - HY16F188_Power : Modify the GPIO initialization(GPIO setting-->Clear GPIO Flag-->judgement GPIO INT)