HY16F3910 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.

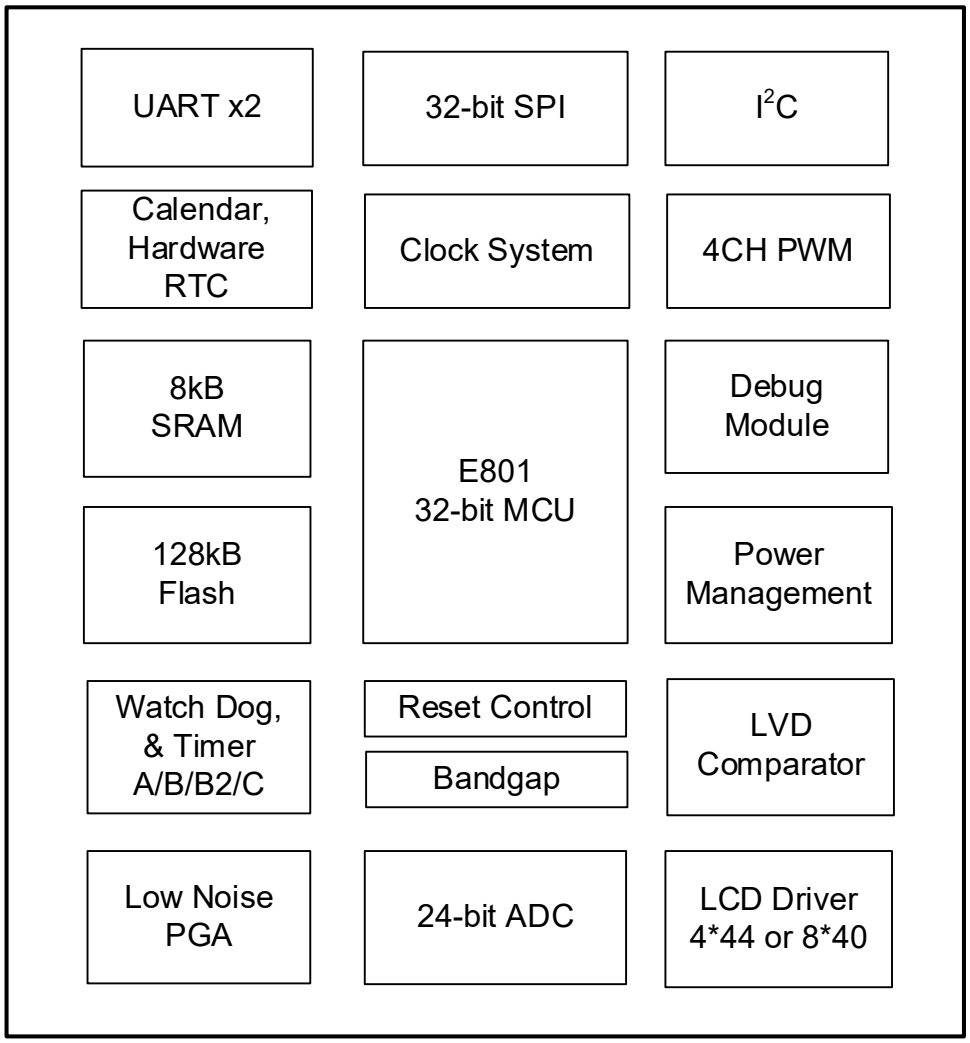
1. Document Description

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 HY16F3910, of which can be grouped into four categories:

(1) Digital IP : TimerA/TimerB/timerB2/TimerC/WDT/PWM/Hardware RTC/GPIO
(2) Analog IP : ADC/LCD
(3) Communication IP : Hardware 32-bit SPI/ Hardware UART/ Hardware UART 2/ Hardware I2C
(4) Peripheral IP : Power Management

2. IC Description

Basic description of each HY16F3910 IP.



(01) Adopting Andes Technology 32-bit CPU core, E801 processor.

(02) Voltage operation range: 2.0~5.5V(No analog power enable condition), and temperature operation range: -40℃~85℃.

(03) Support external 16MHz crystal oscillator or internal 32MHz RC oscillator,

(04) Program memory: 128K-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

(8) 16-bit Timer A

(9) 16-bit Timer B/B2 module has PWM waveform generating function

(10) 16-bit Timer C module has digital capture function

(11) Hardware serial communication 32-bit SPI/I2C/UART*2 module

(12) Hardware RTC clock function module

(13) LCD Driver

※Interrupt control system

Interrupt Vector Address	Vector	Interrupt Function
INT Base Address + 0X00 (I2C/UART/SPI interface)	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 (LVD/BOR2)	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)
INT Base Address + 0X18 (UART2)	HW7	void HW7_ISR(void)
INT Base Address + 0X1C (TMB2)	HW8	void HW8_ISR(void)
INT Base Address + 0X20 (PT3)	HW9	void HW9_ISR(void)

3. Digital IP (Timer)

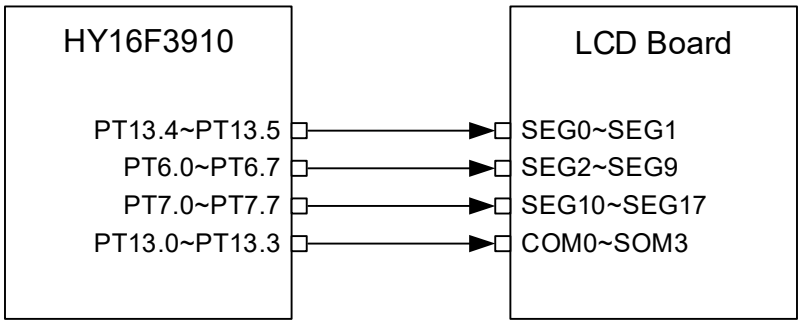
3.1. Example Name

HY16F3910_Timer

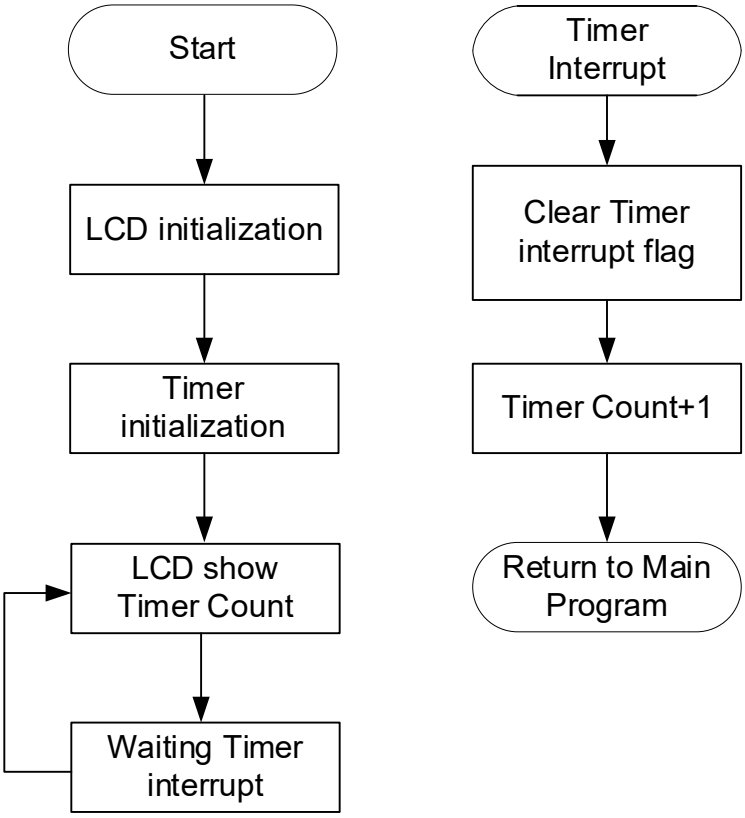
3.2. Example Description

(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. System Description



3.4. Software Flowchart



3.5. 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 : HY16F3910_Timer
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.5
 * MCU Device     :
 * Description    :
```

```
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 *
 *     HY16F3910                        LCD Board(HY10000-AM01)
 *----------------                 ----------------
 *                          |             |
 *    PT13.4~PT13.5 | -----> | SEG0~SEG1
 *       PT6.0~PT6.7 | -----> | SEG2~SEG9
 *       PT7.0~PT7.7 | -----> | SEG10~SEG17
 *    PT13.0~PT13.3 | -----> | COM0~COM3
 *                          |             |
 *----------------                 ----------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "DrvPMU.h"
#include "DrvTimer.h"
#include "Display.h"
#include "main.h"
#include "System.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMB2done:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_WDTdone:1;
      unsigned REV:1;
   };
} MCUSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ


#define    TMATEST
```

```
//#define   TMBTEST
//#define   TMCTEST

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS   MCUSTATUSbits;
unsigned int TimerA_count, TimerB_count, TimerC0_count, TimerC1_count;

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

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
        DrvCLOCK_EnableHighOSC(E_INTERNAL,100);       // Select HSRC
      #if defined(HAO_4MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                  //Select internal 4.147MHz, and calibration
4.147MHz
        DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);              //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
      #elif defined(HAO_32MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                 //Select internal 31.795MHz, and calibration
31.795MHz
        DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);              //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
      #endif
#endif
#if defined(HSXT)
  DrvCLOCK_SelectOHS_HS(1);     //0:HSXT<4MHZ 1:HSXT>4MHZ
  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

  TimerA_count=0;
  TimerB_count=0;
  TimerC0_count=0;
  TimerC1_count=0;
  DisplayInit();
  ClearLCDframe();

  DisplayHYcon();
  Delay(10000);

#if defined(TMATEST)
  DrvTMA_Open(15,1);                                  //Timer A Overflow
                                                      //15:taclk/65536/32;TMRDV=÷32
                                                      //1: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(1,3);                          //Timer B Prescaler 1
                                                      //1: HS_CK,clock source.
                                                      //3: clock divider.÷8
    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(1,1);                          //Timer B Prescaler 1
                                                      //1: HS_CK,clock source.
                                                      //1: clock divider.÷2
    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:÷16384 1:Rising-edge trigger
    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

    MCUSTATUSbits._byte = 0;
    SYS_EnableGIE(4,0x3BF);                           //Enable GIE(Global Interrupt)

    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;
      }
    }

}

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

```
/* 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)
{
  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  : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
```

```
/* Description    : PT3 interrupt Service Routine (HW9).                  */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark         :                                                       */
/*------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                             */
/* Description    : Exception Service Routines.                           */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark         :                                                       */
/*------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                               */
/* Description    : Exception Service Routines.                           */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark         :                                                       */
/*------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*------------------------------------------------------------------------*/
/* Software Delay Subroutines                                             */
/*------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

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

4. Digital IP (WDT)

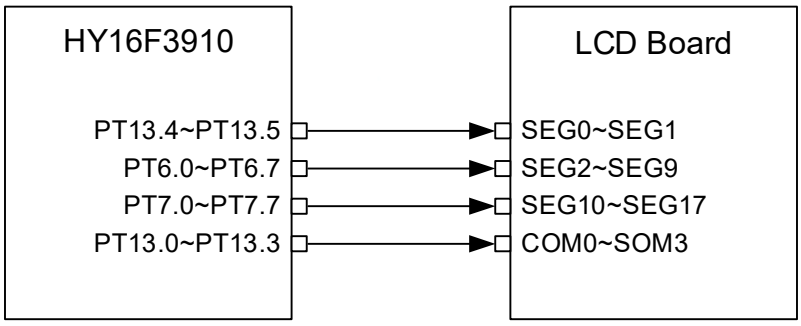
4.1. Example Name

HY16F3910_WDT

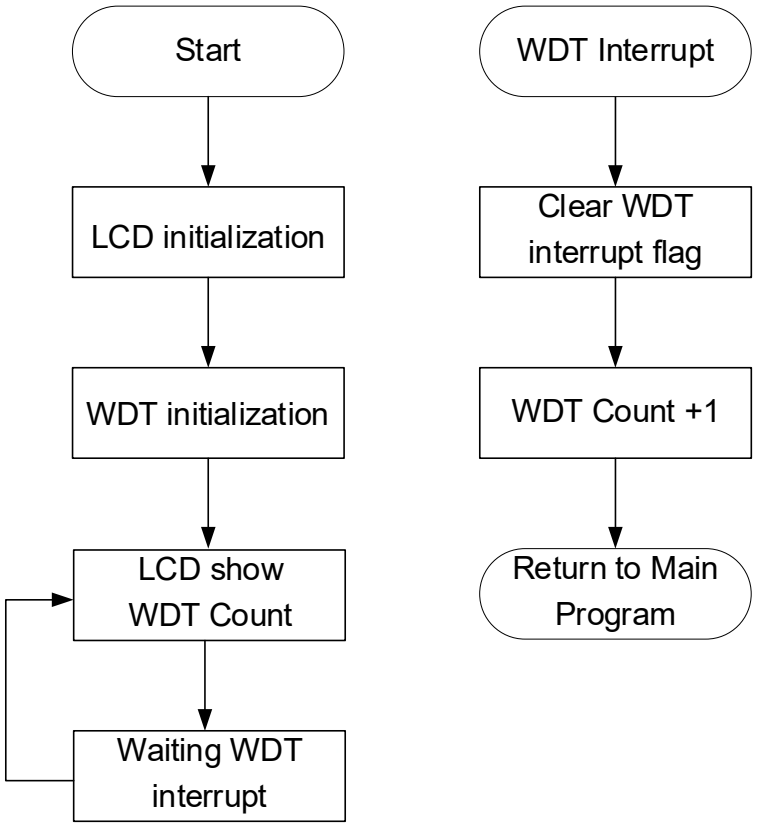
4.2. Example Description

(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. System Description



4.4. Software Flowchart



4.5. 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 : HY16F3910_WDT
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2020/03/22
 * Created by    :
 *
 * Program Description:
```

```
 *
 *     HY16F3910                        LCD Board(HY10000-AM01)
 *----------------                 ----------------
 *                        |             |
 *    PT13.4~PT13.5 | -----> | SEG0~SEG1
 *       PT6.0~PT6.7 | -----> | SEG2~SEG9
 *       PT7.0~PT7.7 | -----> | SEG10~SEG17
 *    PT13.0~PT13.3 | -----> | COM0~COM3
 *                        |             |
 *----------------                ----------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "DrvTimer.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvPMU.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMB2done:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_WDTdone:1;
      unsigned REV:1;
   };
} MCUSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ


/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS    MCUSTATUSbits;
unsigned int WDT_count;
```

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

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
      DrvCLOCK_EnableHighOSC(E_INTERNAL,100);       // Select HSRC
    #if defined(HAO_4MHZ)
      DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                  //Select internal 4.147MHz, and calibration
4.147MHz
      DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);              //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
    #elif defined(HAO_32MHZ)
      DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                 //Select internal 31.795MHz, and calibration
31.795MHz
      DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);              //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
    #endif
#endif
#if defined(HSXT)
  DrvCLOCK_SelectOHS_HS(1);    //0:HSXT<4MHZ 1:HSXT>4MHZ
  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

  DisplayInit();
  ClearLCDframe();

  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
  WDT_count=0;
  MCUSTATUSbits._byte = 0;
  SYS_EnableGIE(4,0x3BF);                //Enable GIE(Global Interrupt)

  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)
```

```
{

}
/*---------------------------------------------------------------------------*/
/* 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))
  {
      DrvWDT_ClearWDT();
      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  : LVD & BOR2 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: HW7_ISR()                                                */
/* Description  : UART2 interrupt Service Routine (HW7).                   */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                */
/* Description  : TMB2 interrupt Service Routine (HW8).                    */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                */
/* Description  : PT3 interrupt Service Routine (HW9).                     */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                              */
/* Description  : Exception Service Routines.                              */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                */
/* Description  : Exception Service Routines.                              */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
```

```
/* Remark          :                                                          */
/*-------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}

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

5. Digital IP (WDT Reset)

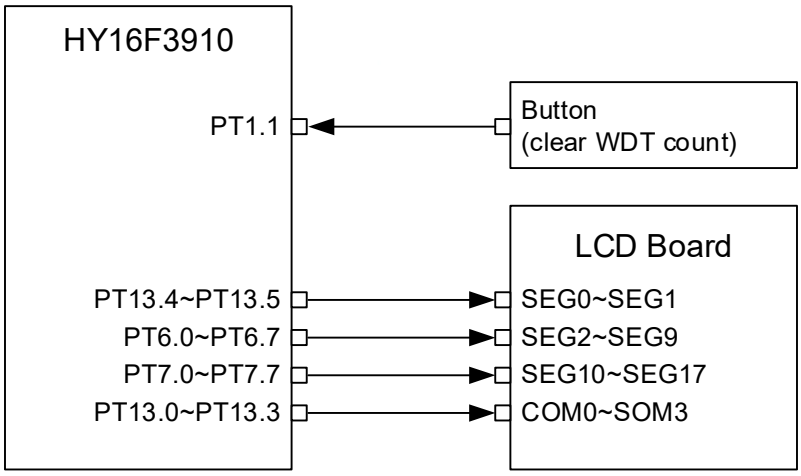
5.1. Example Name

HY16F3910_WDT_Reset

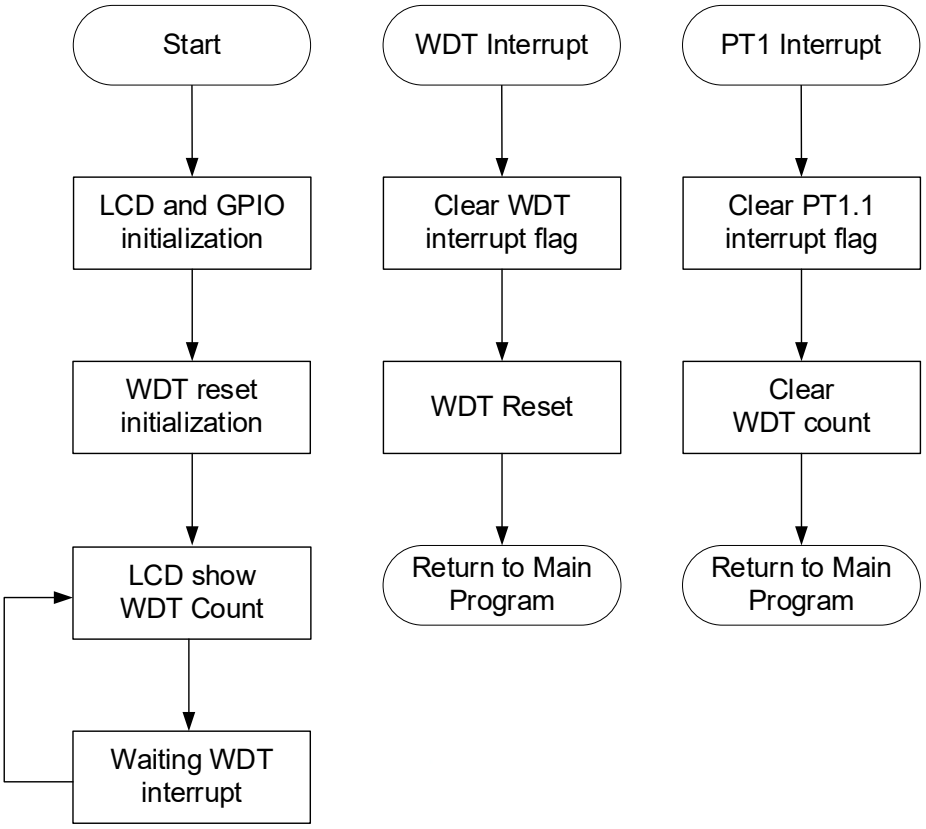
5.2. Example Description

(1) WDT Reset tutorial
(2) n 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 2500~3400, 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 PT1.2 button to clear WDT counter register. It also set WDT Count=0, restart to do variable "WDT Count" +1.
(5) note : When enable WDT Reset mode, user can't switch to WDT Timer mode.

5.3. System Description



5.4. Software Flowchart



5.5. 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 : HY16F3910_WDT_Reset
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.4
 * MCU Device     :
 * Description    :
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 *
 *       HY16F3910
 *-----------------              ----------------
 *                      |           |
 *                PT1.1 | <----- | button1 (Clear WDT count)
 *                      |           |
 *                      |             ----------------
 *                      |
 *                      |               LCD Board(HY10000-AM01)
 *                      |             ----------------
 *                      |           |
 *     PT13.4~PT13.5 | -----> | SEG0~SEG1
 *        PT6.0~PT6.7 | -----> | SEG2~SEG9
 *        PT7.0~PT7.7 | -----> | SEG10~SEG17
 *     PT13.0~PT13.3 | -----> | COM0~COM3
 *                      |           |
 *                      |             ----------------
 *-----------------
 *
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "DrvTimer.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvPMU.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
```

```
    {
        unsigned b_TMAdone:1;
        unsigned b_TMBdone:1;
        unsigned b_TMB2done:1;
        unsigned b_TMC0done:1;
        unsigned b_TMC1done:1;
        unsigned b_RTCdone:1;
        unsigned b_WDTdone:1;
        unsigned REV: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 HAO_4MHZ
#define HAO_32MHZ

#define KEY_PORT E_PT1
#define KEYIN0 BIT1
#define KEYIN0_PIN 1
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS    MCUSTATUSbits;
PTINTSTATUS    PT1INTSTATUSbits;
unsigned int WDT_count;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
    unsigned int WDT_count=0;
    DrvPMU_VDD15Trim();    // Make sure VDD15 Voltage,before change OSC setting
    DrvCLOCK_EnableHighOSC(E_INTERNAL,100);         // Select HSRC
#if defined(HAO_4MHZ)
    DrvCLOCK_SelectIHOSC(E_HAO_4M);                    //Select internal 4.147MHz
    DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);     //ENMCD=11b,MCU Clock/1, CPU CLOCK IS 'HS_CK/1'
    DrvCLOCK_CalibrateHAO(TRIM_HAO4MHZ);               //Calibration 4.147MHz
#elif defined(HAO_32MHZ)
    DrvCLOCK_SelectIHOSC(E_HAO_32M);                   //Select internal 31.795MHz
    DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);     //ENMCD=00b,MCU Clock/2, CPU CLOCK IS 'HS_CK/2'
    DrvCLOCK_CalibrateHAO(TRIM_HAO32MHZ);              //Calibration 31.795MHz
#endif

    DisplayInit();
```

```
    ClearLCDframe();

    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_Open(KEY_PORT,KEYIN0,E_IO_INPUT);               //set PT1.1 INPUT
    DrvGPIO_Open(KEY_PORT,KEYIN0,E_IO_PullHigh);            //enable PT1.1 pull high R
    DrvGPIO_Open(KEY_PORT,KEYIN0,E_IO_IntEnable);           //PT1.1 interrupt enable
    DrvGPIO_IntTrigger(KEY_PORT,KEYIN0,E_N_Edge);           //PT1.1 interrupt trigger method is negative edge
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN0);                  //clear PT1 interrupt flag

    MCUSTATUSbits._byte = 0;
    PT1INTSTATUSbits._byte = 0;
    SYS_EnableGIE(4,0x3BF);                                 //Enable GIE(Global Interrupt)

    while(1)                                                //Wait for Interrupt
    {
      for(WDT_count=0;WDT_count<999999;WDT_count++)
      {
        LCD_DATA_DISPLAY(WDT_count);                        //WDT Reset occur on about WDT_count=4000~4500, it
depends on the HAO frequency
        Delay(1000);
        if(PT1INTSTATUSbits.b_PTINT1done)                   //if PT1.1 low
        {
          WDT_count=0;                                      //Set WDT_count=0,
          DrvWDT_ClearWDT();                                //WDT re-count again, start from 0 to count
          PT1INTSTATUSbits.b_PTINT1done=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)
{

}
/*---------------------------------------------------------------------------*/
/* 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  : LVD & BOR2 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_PTINT1done=1;
  }
  DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN0);          //clear PT1.1 interrupt flag
}

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

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{
```

```
}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
/* Description  : PT3 interrupt Service Routine (HW9).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
      asm("NOP");
}
/*---------------------------------------------------------------------------*/
/* Power command                                                             */
/*---------------------------------------------------------------------------*/
void enterIdleMode(void)
{
   SOCSTATUSL = 0x1010; // IDLE = 1!!
   asm("STANDBY wake_grant");
}

void enterSleepMode(void)
{
   SOCSTATUSL = 0x1000; // IDLE = 0
   asm("STANDBY wake_grant");
}

void enterWaitMode(void)
{
   SOCSTATUSL = 0x1000; // Clear IDLE
   asm("STANDBY no_wake_grant");
```

```
}

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

6. Digital IP (RTC)

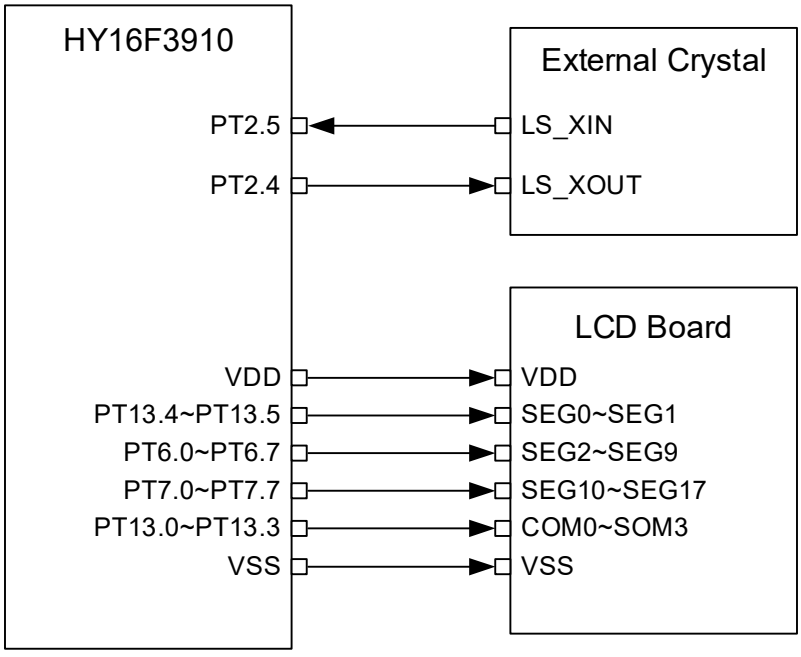
6.1. Example Name

HY16F3910_RTC

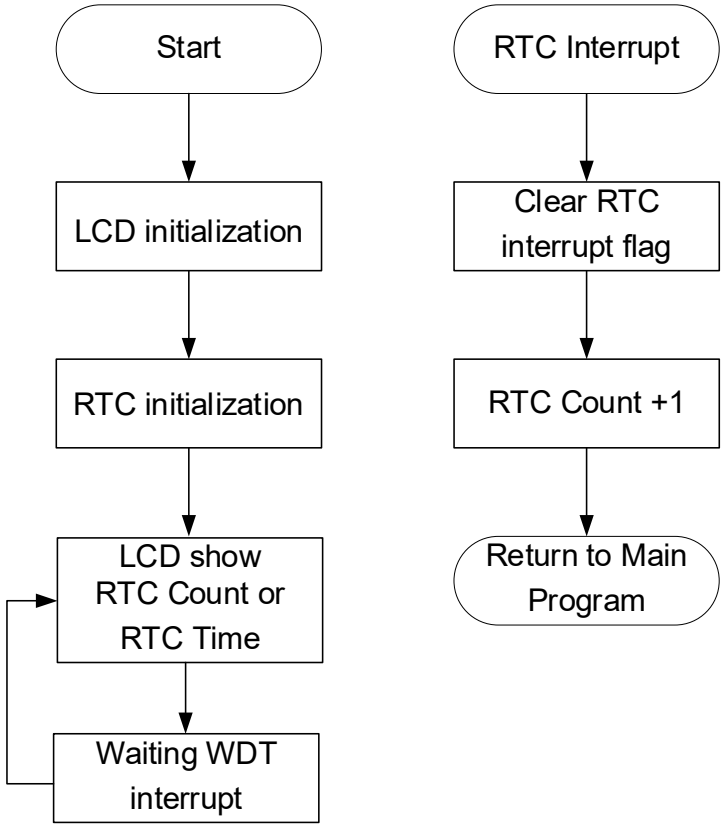
6.2. Example Description

(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. System Description



6.4. Software Flowchart



6.5. 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 : HY16F3910_RTC
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.5
 * MCU Device     :
 * Description    :
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 *
 *     HY16F3910                    External crystal
```

```
 *----------------              ----------------
 *                    |           |
 *             PT2.5 | <----- | LS_XIN
 *             PT2.4 | -----> | LS_XOUT
 *                    |           |
 *                    |             ----------------
 *                    |
 *                    |               LCD Board(HY10000-AM01)
 *                    |             ----------------
 *                    |           |
 *   PT13.4~PT13.5 | -----> | SEG0~SEG1
 *      PT6.0~PT6.7 | -----> | SEG2~SEG9
 *      PT7.0~PT7.7 | -----> | SEG10~SEG17
 *   PT13.0~PT13.3 | -----> | COM0~COM3
 *                    |           |
 *                    |             ----------------
 *----------------
 *
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "DrvPMU.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvRTC.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
   char   _byte;
   struct
   {
      unsigned b_TMAdone:1;
      unsigned b_TMBdone:1;
      unsigned b_TMB2done:1;
      unsigned b_TMC0done:1;
      unsigned b_TMC1done:1;
      unsigned b_RTCdone:1;
      unsigned b_WDTdone:1;
      unsigned REV:1;
   };
} MCUSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//OSC
#define HSXT    //External 16MHz
//#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ
```

```
//#define   RTC_counter_show
#define   TimerDtata_show

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS   MCUSTATUSbits;
unsigned int sec,min,hour,week,day,month,year ;
unsigned int RTC_counter;
unsigned int TimerDtata;

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void InitalRTC(void);
int ComputeWeek(int TempYear, int TempMonth, int TempDay);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  S_DRVRTC_TIME_DATA_T sCurTime;      //Setting Start
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
        DrvCLOCK_EnableHighOSC(E_INTERNAL,100);        // Select HSRC
      #if defined(HAO_4MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                  //Select internal 4.147MHz, and calibration
4.147MHz
        DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                  //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
      #elif defined(HAO_32MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                  //Select internal 31.795MHz, and
calibration 31.795MHz
        DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                  //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
      #endif
#endif
#if defined(HSXT)
  DrvCLOCK_SelectOHS_HS(1);    //0:HSXT<4MHZ 1:HSXT>4MHZ
  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

  RTC_counter=0;
  TimerDtata=0;
  DisplayInit();
  ClearLCDframe();
  InitalRTC();
  MCUSTATUSbits._byte = 0;
  SYS_EnableGIE(4,0x3BF);                  //Enable GIE(Global Interrupt)

  while(1)                                 //Wait for Interrupt
  {
    if(MCUSTATUSbits.b_RTCdone==1)
    {
    #if defined(TimerDtata_show)
      DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
      TimerDtata=sCurTime.u32cSecond+sCurTime.u32cMinute*100+sCurTime.u32cHour*10000;
```

```
            LCD_DATA_DISPLAY(TimerDtata);
            sec=sCurTime.u32cSecond;
            min=sCurTime.u32cMinute;
            hour=sCurTime.u32cHour;
            week=sCurTime.u32cDayOfWeek;
            day=sCurTime.u32cDay;
            month=sCurTime.u32cMonth;
            year=sCurTime.u32Year;
        #elif defined(RTC_counter_show)
            LCD_DATA_DISPLAY(RTC_counter);
        #endif
            MCUSTATUSbits.b_RTCdone=0;
        }
        //DrvCLOCK_LPOKeep(1);
        //SYS_LowPower(SYS_SleepMode);
    }

    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)
{
    if(DrvRTC_ReadIntFlag())
    {
        if(DrvRTC_ReadState()&0x4)     //check PTF flag
        {
            DrvRTC_ClearState(E_DRVRTC_PERIODIC_FLAG);
            RTC_counter++;

        }
          if(DrvRTC_ReadState()&0x1)     //check TAF flag
        {
            DrvRTC_ClearState(E_DRVRTC_ALARM_FLAG);

        }
          DrvRTC_ClearIntFlag();
        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  : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

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

```
/* Function Name: HW8_ISR()                                                    */
/* Description  : TMB2 interrupt Service Routine (HW8).                        */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*-----------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                    */
/* Description  : PT3 interrupt Service Routine (HW9).                         */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*-----------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                  */
/* Description  : Exception Service Routines.                                  */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-----------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                    */
/* Description  : Exception Service Routines.                                  */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-----------------------------------------------------------------------------*/
/* Function Name: InitalRTC()                                                  */
/* Description  : RTC Initialization Subroutines.                              */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void InitalRTC()
{
   S_DRVRTC_TIME_DATA_T sCurTime;              //Setting Start
   DrvCLOCK_EnableLowOSC(E_EXTERNAL,130000);
```

```
    DrvRTC_ClockSource(E_EXTERNAL_CLOCK);

    //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
}

/*---------------------------------------------------------------------------*/
/* Compute Week Subroutines                                                  */
/*---------------------------------------------------------------------------*/
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);
}

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

7. Digital IP (PWM)

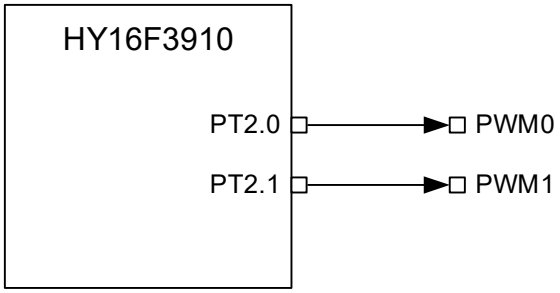
7.1. Example Name

HY16F3910_PWM

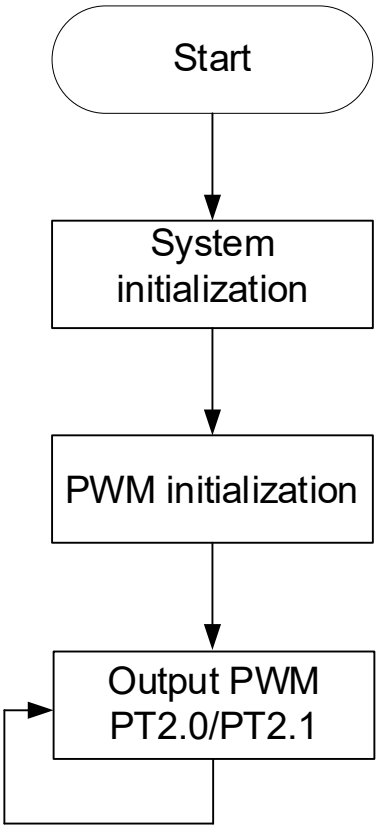
7.2. Example Description

(1) PWM tutorial.
(2) Set HAO 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. System Description



7.4. Software Flowchart



7.5. Software Flowchart

```
/*******************************************************************************
 *
 * 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 : HY16F3910_PWM
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2022/9/20
 * Created by    :
 *
 * Program Description:
 *
```

```
 *   HY16F3910                          PWM
 *----------------              ----------------
 *                       |            |
 *               PT2.0 | -----> | PWM0
 *               PT2.1 | -----> | PWM1
 *                       |            |
 *                       |              ----------------
 *                       |
 *----------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvPMU.h"
#include "DrvTimer.h"
#include "DrvGPIO.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ

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


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

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
   DrvPMU_VDD15Trim();    // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
          DrvCLOCK_EnableHighOSC(E_INTERNAL,100);        // Select HSRC
       #if defined(HAO_4MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
```

```
            DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                    //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
        #elif defined(HAO_32MHZ)
            DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                    //Select internal 31.795MHz, and
calibration 31.795MHz
            DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                    //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
        #endif
#endif
#if defined(HSXT)
   DrvCLOCK_SelectOHS_HS(1);     //0:HSXT<4MHZ 1:HSXT>4MHZ
   DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

   SYS_EnableGIE(4,0x3BF);    //Enable GIE(Global Interrupt)

   DrvPWM0_Open(E_PWMA,1,2);                                    //PWM0 Enable Port 2.0 =PWMO0, Port 2.1
=PWMO1(Set PWM0=PT2.0)
   DrvPWM1_Open(E_PWMA,1,2);                                    //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(1,0);
   DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF);    //TMB Overflow 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  : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{
```

```
}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
/* Description   : PT3 interrupt Service Routine (HW9).                      */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

8. Digital IP(Flash)

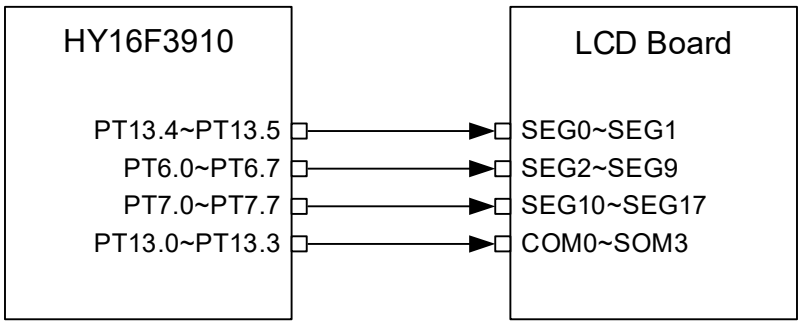
8.1. Example Name

HY16F3910_Flash

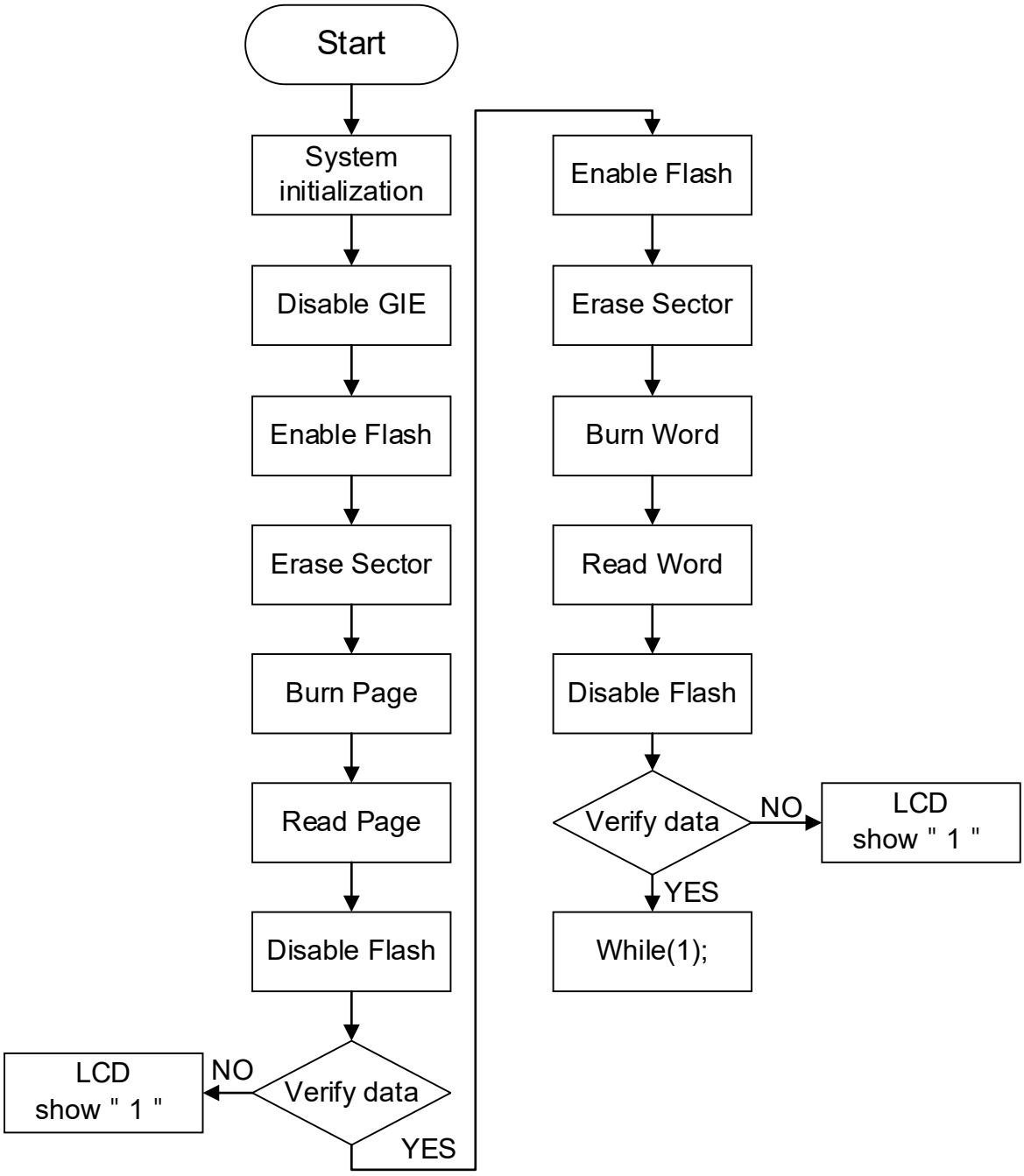
8.2. Example Description

(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, LCD display"1" and 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 : Before Flash programming and erasing, the Flash enable command must be executed. After programming and erasing, the Flash disable command must be executed to avoid unexpected actions.
Note2 : VDD5V have to more than 1.8V, it can prevent program burn error when executing Flash burn function.

8.3. System Description



8.4. Software Flowchart


Start
System initialization
Disable GIE
Enable Flash
Erase Sector
Burn Page
Read Page
Disable Flash
Verify data
NO
LCD show " 1 "
YES
Enable Flash
Erase Sector
Burn Word
Read Word
Disable Flash
Verify data
NO
LCD show " 1 "
YES
While(1);

8.5. 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 : HY16F3910_Flash
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2022/03/22
 * Created by    :
 *
 * Program Description:
 *
 *    HY16F3910                    LCD Board(HY10000-AM01)
 *----------------              ----------------
 *                      |            |
 *   PT13.4~PT13.5 | -----> | SEG0~SEG1
 *      PT6.0~PT6.7 | -----> | SEG2~SEG9
 *      PT7.0~PT7.7 | -----> | SEG10~SEG17
 *   PT13.0~PT13.3 | -----> | COM0~COM3
 *                      |            |
 *----------------              ----------------
 *
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvLCD.h"
#include "DrvClock.h"
#include "DrvPMU.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "BootFunc.h"
#include "DrvFlash.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
```

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


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//#define HAO_4MHZ
#define HAO_32MHZ

/*---------------------------------------------------------------------------*/
/* Data Flash                                                                */
/*---------------------------------------------------------------------------*/
const unsigned char DefaultData[256] __attribute__ ((section ("DATA_EARE0"))) =
{
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F,
0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F,
0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F,
0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F,
0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F,
0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F,
0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F,
0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E, 0x8F,
0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F,
0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF,
0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF,
0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF,
0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF,
0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF,
0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
};   //0xAFC00~0xAFC
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  unsigned char index,error;
  unsigned int BufferTx[32];
  int BufferRx[32];
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
  DrvCLOCK_EnableHighOSC(E_INTERNAL,100);        // Select HSRC
  #if defined(HAO_4MHZ)
  DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
  DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                  //ENMCD=01b,MCU Clock/1, CPU CLOCK IS
'HS_CK/1'
  #elif defined(HAO_32MHZ)
  DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                   //Select internal 31.795MHz, and calibration
31.795MHz
  DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                  //ENMCD=00b,MCU Clock/2, CPU CLOCK IS
'HS_CK/2'
```

```
#endif

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

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

//BurnPage
//demo1
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
}
ISP_FUNC_ROMP->FlashOpEn();                //Flash write enable.
ISP_FUNC_ROMP->SectorErase(0xAFC00);       //Flash erase 1KB sector, erase 0xAFC00~0xAFFFF
ISP_FUNC_ROMP->BurnPage(0xAFC00,BufferTx,32);  //Flash page write, write address 0xAFC00~0xAFC7F
ReadPage(0xAFC00,BufferRx);                //Flash page read, read address 0xAFC00~0xAFC7F
ISP_FUNC_ROMP->FlashOpDis();               //Flash write disable.

//verify the ROM_BurnPage function, if fail to show "1" on the LCD Display
for(index=0;index<32;index++)
{
   if(BufferTx[index]!=BufferRx[index])
   {
      LCD_DATA_DISPLAY(1);
      while(1);
   }
}

//demo2
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
}
ISP_FUNC_ROMP->FlashOpEn();                //Flash write enable.
//ISP_FUNC_ROMP->SectorErase(0xAFC80);     //Flash erase 1KB sector, erase 0xAFC00~0xAFFFF
ISP_FUNC_ROMP->BurnPage(0xAFC80,BufferTx,32);  //Flash page write, write address 0xAFC80~0xAFCFF
ReadPage(0xAFC80,BufferRx);                //Flash page read, read address 0xAFC80~0xAFCFF
ISP_FUNC_ROMP->FlashOpDis();               //Flash write disable.

//verify the ROM_BurnPage function, if fail to show "1" on the LCD Display
for(index=0;index<32;index++)
{
   if(BufferTx[index]!=BufferRx[index])
   {
      LCD_DATA_DISPLAY(1);
      while(1);
   }
}

//BurnWord
//demo1
ISP_FUNC_ROMP->FlashOpEn();                //Flash write enable.
```

```
    //ISP_FUNC_ROMP->SectorErase(0xAFC00);                    //Flash erase 1KB sector
    error=ISP_FUNC_ROMP->BurnWord(0xAFD00,0x12345678);  //Flash word write, write address 0xAFD00
    if(error==0)                                            //if Error=0, it means BurnWord success
    {
      BufferRx[0]=ReadWord(0xAFD00);
      ISP_FUNC_ROMP->FlashOpDis();                          //Flash write disable.
    }
    if(error==3)                                            //if Error=3, it means BurnWord failure
    {
      LCD_DATA_DISPLAY(1);
      while(1);
    }

    //demo2
    ISP_FUNC_ROMP->FlashOpEn();                             //Flash write enable.
    //ISP_FUNC_ROMP->SectorErase(0xAFC00);                    //Flash erase 1KB sector
    error=ISP_FUNC_ROMP->BurnWord(0xAFD04,0xFFFFFF50);  //Flash word write, write address 0xAFD04
    if(error==0)                                            //if Error=0, it means BurnWord success
    {
      BufferRx[1]=ReadWord(0xAFD04);
      ISP_FUNC_ROMP->FlashOpDis();                          //Flash write disable.
    }
    if(error==3)                                            //if Error=3, it means BurnWord failure
    {
      LCD_DATA_DISPLAY(1);
      while(1);
    }


    //demo3
    ISP_FUNC_ROMP->FlashOpEn();                             //Flash write enable.
    //ISP_FUNC_ROMP->SectorErase(0xAFC00);                    //Flash erase 1KB sector
    error=ISP_FUNC_ROMP->BurnWord(0xAFD08,0xABCD5678);  //Flash word write, write address 0xAFD08
    if(error==0)                                            //if Error=0, it means BurnWord success
    {
      BufferRx[2]=ReadWord(0xAFD08);
      ISP_FUNC_ROMP->FlashOpDis();                          //Flash write disable.
    }
    if(error==3)                                            //if Error=3, it means BurnWord failure
    {
      LCD_DATA_DISPLAY(1);
      while(1);
    }

    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  : LVD & BOR2 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: HW7_ISR()                                                  */
```

```
/* Description    : UART2 interrupt Service Routine (HW7).                    */
/* Arguments      : None.                                                     */
/* Return Value : None.                                                       */
/* Remark         :                                                           */
/*----------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*----------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                   */
/* Description    : TMB2 interrupt Service Routine (HW8).                     */
/* Arguments      : None.                                                     */
/* Return Value : None.                                                       */
/* Remark         :                                                           */
/*----------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*----------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                   */
/* Description    : PT3 interrupt Service Routine (HW9).                      */
/* Arguments      : None.                                                     */
/* Return Value : None.                                                       */
/* Remark         :                                                           */
/*----------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*----------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                 */
/* Description    : Exception Service Routines.                               */
/* Arguments      : None.                                                     */
/* Return Value : None.                                                       */
/* Remark         :                                                           */
/*----------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*----------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                   */
/* Description    : Exception Service Routines.                               */
/* Arguments      : None.                                                     */
/* Return Value : None.                                                       */
/* Remark         :                                                           */
/*----------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*----------------------------------------------------------------------------*/
```

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

9. Digital IP(GPIO)

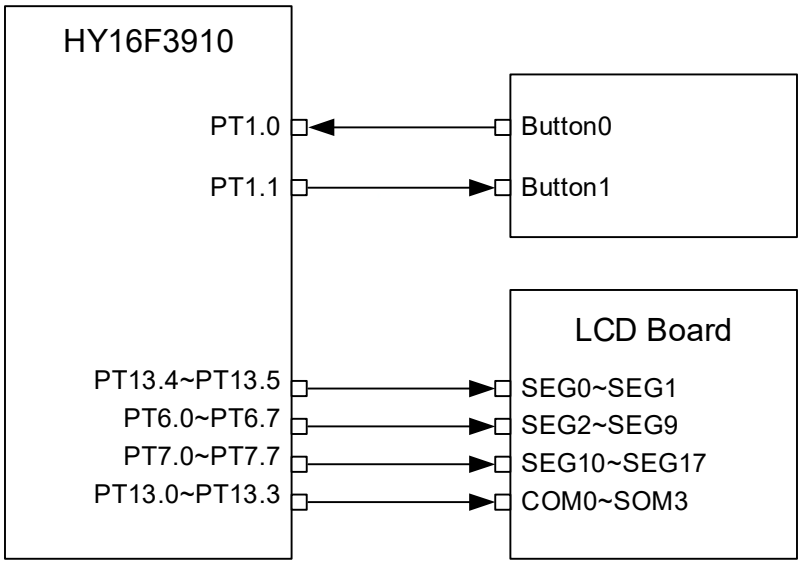
9.1. Example Name

HY16F3910_GPIO

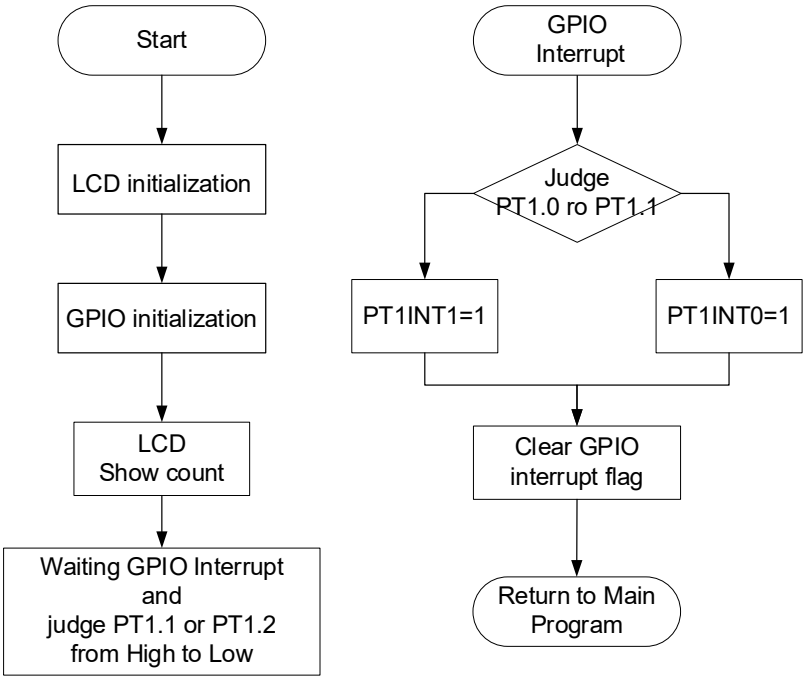
9.2. Example Description

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

9.3. System Description



9.4. Software Flowchart



9.5. 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 : HY16F3910_GPIO
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.5
 * MCU Device     :
 * Description    :
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 *
```

```
 *      HY16F3910
 *----------------               ----------------
 *                      |            |
 *              PT1.0 | <----- | button0
 *              PT1.1 | <----- | button1
 *                      |            |
 *                      |              ----------------
 *                      |
 *                      |                LCD Board(HY10000-AM01)
 *                      |              ----------------
 *                      |            |
 *   PT13.4~PT13.5 | -----> | SEG0~SEG1
 *      PT6.0~PT6.7 | -----> | SEG2~SEG9
 *      PT7.0~PT7.7 | -----> | SEG10~SEG17
 *   PT13.0~PT13.3 | -----> | COM0~COM3
 *                      |            |
 *                      |              ----------------
 *----------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvPMU.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvGPIO.h"

/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile 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                                                               */
/*---------------------------------------------------------------------------*/
//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
```

```
#define HAO_32MHZ

#define KEY_PORT E_PT1
#define KEYIN0 BIT0
#define KEYIN1 BIT1
#define KEYIN0_PIN 0
#define KEYIN1_PIN 1

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

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

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
   DrvPMU_VDD15Trim();    // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
          DrvCLOCK_EnableHighOSC(E_INTERNAL,100);         // Select HSRC
       #if defined(HAO_4MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
          DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                  //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
       #elif defined(HAO_32MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                   //Select internal 31.795MHz, and
calibration 31.795MHz
          DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                  //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
       #endif
#endif
#if defined(HSXT)
   DrvCLOCK_SelectOHS_HS(1);     //0:HSXT<4MHZ 1:HSXT>4MHZ
   DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif
   DisplayInit();
   ClearLCDframe();
   Delay(10000);
   DisplayHYcon();
   Delay(10000);
   DrvGPIO_LCDIOOpen(E_PT6, 0xff, E_IO_OUTPUT);
   DrvGPIO_LCDIOOpen(E_PT13, 0xFF, E_IO_OUTPUT);
   int count=0;
   LCD_DATA_DISPLAY(count);
   DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_INPUT);          //set PT1.0/PT1.1 INPUT
   DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_PullHigh);     //enable PT1.0/PT1.1 pull high R
   DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_IntEnable);   //PT1.0/PT1.1 interrupt enable
   DrvGPIO_IntTrigger(KEY_PORT,KEYIN1|KEYIN0,E_N_Edge);   //PT1.0/PT1.1 interrupt trigger method is negative
edge
   DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);              //clear PT1 interrupt flag
```

```
    PT1INTSTATUSbits._byte = 0;
    SYS_EnableGIE(4,0x3BF);                          //Enable GIE(Global Interrupt)

    while(1)                                         //Wait for Interrupt
    {
      if(PT1INTSTATUSbits.b_PTINT0done)              //if PT1.0 low
      {
        LCD_DATA_DISPLAY(count++);
        PT1INTSTATUSbits.b_PTINT0done=0;
      }
      if(PT1INTSTATUSbits.b_PTINT1done)              //if PT1.1 low
      {
        LCD_DATA_DISPLAY(count--);
        PT1INTSTATUSbits.b_PTINT1done=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  : LVD & BOR2 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)                 //PT1.0
  {
    PT1INTSTATUSbits.b_PTINT0done=1;
  }
  if((PORT_IntFlag&KEYIN1)==KEYIN1)                 //PT1.1
  {
    PT1INTSTATUSbits.b_PTINT1done=1;
  }
  DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN1|KEYIN0);   //Clear PT1.1/PT1.0 interrupt flag
}

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

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
```

```
void HW8_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
/* Description  : PT3 interrupt Service Routine (HW9).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

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

10. Analog IP(ADC)

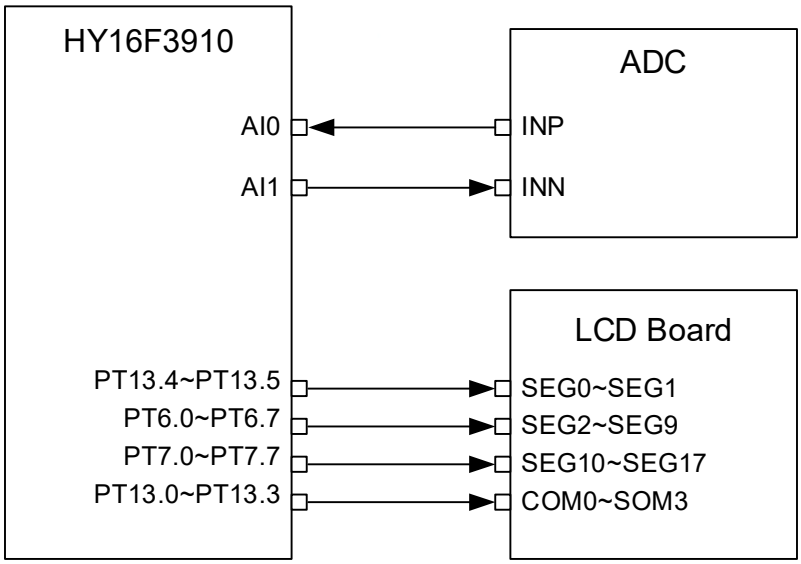
10.1. Example Name

HY16F3910_ADC

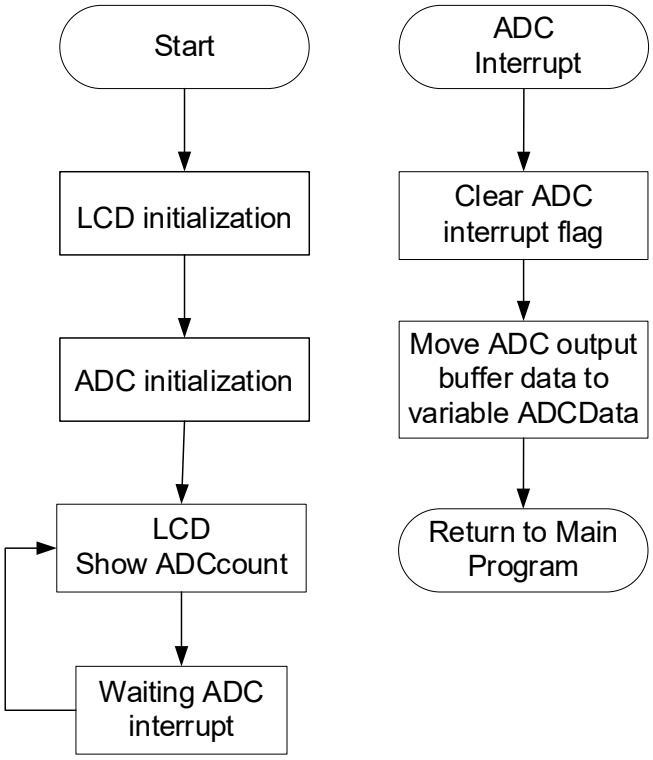
10.2. Example Description

(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"

10.3. System Description



10.4. Software Flowchart



10.5. 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 : HY16F3910_ADC
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.5
 * MCU Device     :
 * Description    :
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 *
 *     HY16F3910                       ADC
 *----------------              ----------------
 *                      |           |
```

```
 *                AI0 | <----- | INP
 *                AI1 | <----- | INN
 *                    |        |
 *                    |          ----------------
 *                    |
 *                    |            LCD Board(HY10000-AM01)
 *                    |          ----------------
 *                    |        |
 *   PT13.4~PT13.5 | -----> | SEG0~SEG1
 *      PT6.0~PT6.7 | -----> | SEG2~SEG9
 *      PT7.0~PT7.7 | -----> | SEG10~SEG17
 *   PT13.0~PT13.3 | -----> | COM0~COM3
 *                    |        |
 *                    |          ----------------
 *----------------
 *
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvPMU.h"
#include "DrvADC.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                                                               */
/*---------------------------------------------------------------------------*/
//OSC
#define HSXT    //External 16MHz
//#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ
```

```
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS    MCUSTATUSbits;
int ADCData;

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void InitalADC(void);
void Delay(unsigned int num);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
        DrvPMU_VDD15Trim();    // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
          DrvCLOCK_EnableHighOSC(E_INTERNAL,100);         // Select HSRC
        #if defined(HAO_4MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
          DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                  //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
        #elif defined(HAO_32MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                   //Select internal 31.795MHz, and
calibration 31.795MHz
          DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                  //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
        #endif
#endif
#if defined(HSXT)
   DrvCLOCK_SelectOHS_HS(1);     //0:HSXT<4MHZ 1:HSXT>4MHZ
   DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

   DisplayInit();
   ClearLCDframe();
   Delay(10000);
   DisplayHYcon();
   Delay(10000);
   InitalADC();
   MCUSTATUSbits._byte = 0;
   SYS_EnableGIE(4,0x3BF);                          //Enable GIE(Global Interrupt)

   while(1)                                         //Wait for Interrupt
   {
      if(MCUSTATUSbits.b_ADCdone)
      {
         LCD_DATA_DISPLAY(ADCData>>16);        //16bits give up.
         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)
{

}

/*-------------------------------------------------------------------------------*/
/* 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    : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
/* Description  : PT3 interrupt Service Routine (HW9).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                */
/* Description  : Exception Service Routines.                                */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
```

```
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
  asm("nop"); //procedure define by customer.
  asm("nop");
  while(1);
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalADC()                                                */
/* Description   : ADC Initialization Subroutines.                           */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void InitalADC(void)
{
  #if defined(HSRC)
    #if defined(HAO_4MHZ)
      DrvADC_ClkEnable(2);                  //Setting ADC CLOCK ADCK=HS_CK/4 & Rising edge is high
    #elif defined(HAO_32MHZ)
      DrvADC_ClkEnable(5);                  //Setting ADC CLOCK ADCK=HS_CK/32 & Rising edge is high
    #endif
  #endif
  #if defined(HSXT)   //16MHz
    DrvADC_ClkEnable(4);                   //Setting ADC CLOCK ADCK=HS_CK/16 & Rising edge is high
  #endif

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

  Delay(0x1000);
  DrvADC_ACM(V12);                        //ACMS=1b, 1.2V
  //adc_00=0x00008080;

  //Set ADC input pin
  DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1); //Set the ADC positive/negative input voltage
source.
  DrvADC_Gain(ADC_PGA_Disable,ADC_ADGN_1); //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
}

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

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

11. Analog IP(LCD)

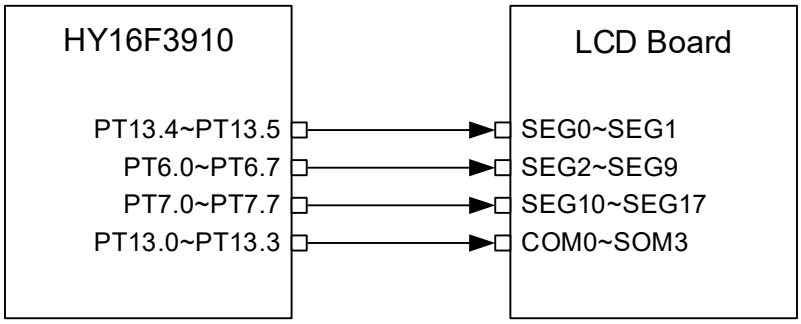
11.1. Example Name

HY16F3910_LCD

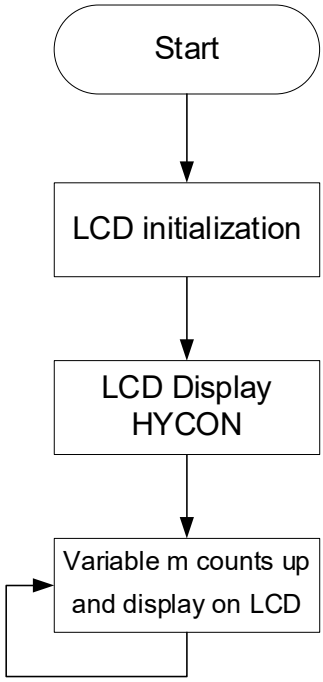
11.2. Example Description

(1) LCD tutorial.
(2) Enable LCD, set VLCD voltage and Duty.
(3) LCD I/O port set as LCD Mode.
(4) LCD display "HYCON"
(5) After LCD display "HYCON", variable "m" counts up and display on LCD.

11.3. System Description



11.4. Software Flowchart



11.5. 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 : HY16F3910_LCD
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2022/03/22
 * Created by    :
 *
 * Program Description:
 *
 *    HY16F3910                    LCD Board(HY10000-AM01)
 *----------------              ----------------
```

```
 *                        |            |
 *   PT13.4~PT13.5 | -----> | SEG0~SEG1
 *      PT6.0~PT6.7 | -----> | SEG2~SEG9
 *      PT7.0~PT7.7 | -----> | SEG10~SEG17
 *   PT13.0~PT13.3 | -----> | COM0~COM3
 *                        |            |
 *-----------------              -----------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvLCD.h"
#include "DrvPMU.h"
#include "DrvClock.h"
#include "Display.h"
#include "main.h"
#include "System.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/


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


/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
   DrvPMU_VDD15Trim();    // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
          DrvCLOCK_EnableHighOSC(E_INTERNAL,100);         // Select HSRC
       #if defined(HAO_4MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
          DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                  //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
       #elif defined(HAO_32MHZ)
          DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                   //Select internal 31.795MHz, and
calibration 31.795MHz
```

```
            DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                    //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
        #endif
#endif
#if defined(HSXT)
   DrvCLOCK_SelectOHS_HS(1);     //0:HSXT<4MHZ 1:HSXT>4MHZ
   DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif


   DisplayInit();
   DrvLCD_LCDBias(E_LCD_BIAS4);
   DrvLCD_LCDBias(E_LCD_BIAS3);

   ClearLCDframe();
   DisplayHYcon();
   Delay(500000);

   unsigned int m;
   for(m=0;m<50000;m++)
   {
      LCD_DATA_DISPLAY(m);
      Delay(65535);
   }
   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  : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{

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

```
/* Function Name: HW9_ISR()                                                    */
/* Description  : PT3 interrupt Service Routine (HW9).                         */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*-----------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                  */
/* Description  : Exception Service Routines.                                  */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-----------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                    */
/* Description  : Exception Service Routines.                                  */
/* Arguments    : None.                                                        */
/* Return Value : None.                                                        */
/* Remark       :                                                              */
/*-----------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-----------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                  */
/*-----------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}

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

12. Communication IP(SPI)

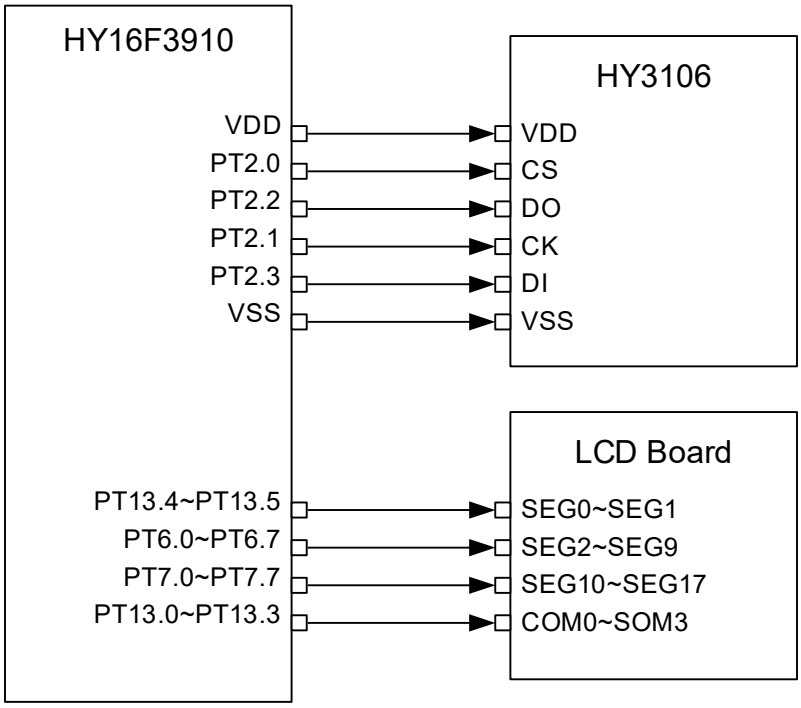
12.1. Example Name

HY16F3910_SPI

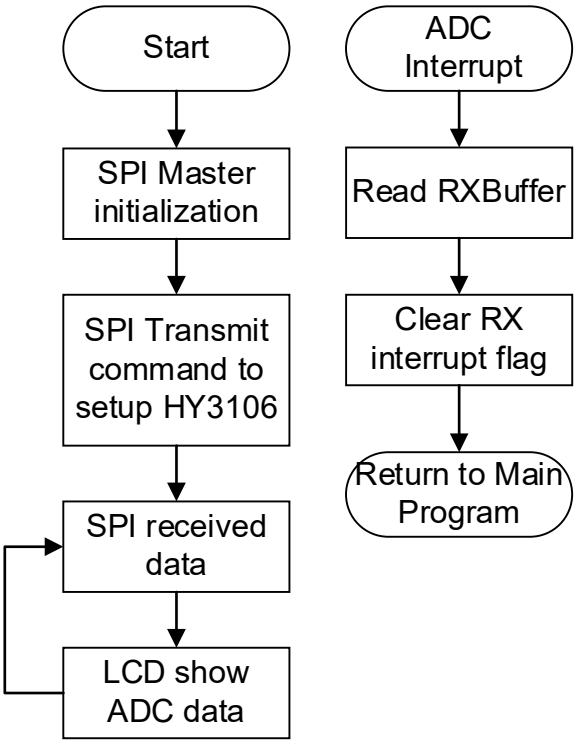
12.2. Example Description

(1) HY16F3910 three-wire SPI Master mode tutorial.
(2)SPI Master communicate with HY3106. Because this example code is three-wire SPI Master mode, so CS(Chip Select) funciton by using GPIO to do control, Set PT2.0=CS function.
(2) The first section of the program HY16F3910 makes ADC settings for HY3106
(3) The second section of the program reads the HY3106 ADC data, and the LCD will display the ADC data

12.3. System Description



12.4. Software Flowchart



12.5. 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 : HY16F3910_SPI
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.5
 * MCU Device     :
 * Description    :
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 * --------------------
 * HY16F3910                    HY3106
 * ---------                  --------
 * VDD          |                |   VDD |
 * SPI_CS     |<--------> |   CS    |
 * SPI_MISO |<--------> |   DO    |
 * SPI_CLK    |<--------> |   CK    |
 * SPI_MOSI |<--------> |   DI    |
 * VSS          |                |   VSS |
 * ---------                  --------
 *******************************************************************************/
/*------------------------------------------------------------------------------*/
/* Includes                                                                     */
/*------------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvSPI32.h"
#include "DrvGPIO.h"
#include "DrvPMU.h"
#include "DrvCLOCK.h"
/*------------------------------------------------------------------------------*/
/* STRUCTURES                                                                   */
/*------------------------------------------------------------------------------*/


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

```
/* DEFINITIONS                                                                    */
/*--------------------------------------------------------------------------------*/
#define   Flash_PORT E_PT2
#define   Flash_CS    BIT0
#define   Flash_SCLK BIT1
#define   Flash_MISO BIT2
#define   Flash_MOSI BIT3
#define   SCS_PIN 0

//OSC
//#define HSXT   //External 16MHz
#define HSRC   //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ

//#define SPI_4WIRE
#define SPI_3WIRE

#define SPI_save_power   //If NO SPI RX Interrupt, enter Idle mode for saving power
/*--------------------------------------------------------------------------------*/
/* Global CONSTANTS                                                               */
/*--------------------------------------------------------------------------------*/
//SPI
unsigned int spi_read_buffer[128];
int ADCData;
unsigned int i;
unsigned int SPI_WRITE8;
unsigned int SPI_WRITE16;
unsigned int SPI_WRITE32;
unsigned int rxIntHappened=0;
unsigned char dummyread=0;
/*--------------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                            */
/*--------------------------------------------------------------------------------*/
void Delay(unsigned int num);
//SPI
void spi_write_8bits(uint32_t spi_address, uint32_t spi_data);
void spi_write_16bits(uint32_t spi_address, uint32_t spi_data);
unsigned int spi_read_8bits(uint32_t spi_address);
unsigned int spi_read_16bits(uint32_t spi_address);
unsigned int spi_read_24bits(uint32_t spi_address);
void InitalSPI(void);
void enterIdleMode(void);   //enter Idle mode, save power
/*--------------------------------------------------------------------------------*/
/* Main Function                                                                  */
/*--------------------------------------------------------------------------------*/
int main(void)
{
  SPI_WRITE8=0;
  SPI_WRITE16=0;
  SPI_WRITE32=0;
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
        DrvCLOCK_EnableHighOSC(E_INTERNAL,100);        // Select HSRC
      #if defined(HAO_4MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
```

```
        DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                  //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
      #elif defined(HAO_32MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                  //Select internal 31.795MHz, and
calibration 31.795MHz
        DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                  //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
      #endif
#endif
#if defined(HSXT)
  DrvCLOCK_SelectOHS_HS(1);    //0:HSXT<4MHZ 1:HSXT>4MHZ
  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

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

  //SPI
  for(i=0;i<=128;i++)
  {
    spi_read_buffer[i]=0;                  //Initial SPI data buffer=0
  }
  InitalSPI();
  SYS_EnableGIE(4,0x3BF);                  //Enable GIE(Global Interrupt)
  DrvSPI32_EnableRxInt();

  //If control HY3106,
  //address 0x00, it is 8bits.
  //address 0x01, it is 16bits.
  //address 0x10, it is 24bits.
  spi_00=0x00000C04; //CPHA=0b, CPOL=1b, write
  spi_write_8bits(0x00,0xF0);   //0xF0:INOSC=1b, LDO=11b, ENLDO=1b, REFOS=0b, SDOH=0b, CH=0b, TS=0b

  spi_00=0x00000C0C; //CPHA=1b, CPOL=1b, read
  spi_read_buffer[0]=spi_read_8bits(0x80);   //If correct, it is 0xF0.

  spi_00=0x00000C04; //CPHA=0b, CPOL=1b
  spi_write_16bits(0x10,0x0035);   //0x0035:DCSET=0000b, INX=00b, ADGN=00b=x1, PGA=001b=x8, FRb=1b=x1/2,
OSR=01b=80sps, ENRB=0b, ADCEN=1b

  spi_00=0x00000C0C; //CPHA=1b, CPOL=1b,
  spi_read_buffer[1]=spi_read_16bits(0x90); //If correct, it is 0x35

  while(1)
  {
    #if defined(HAO_4MHZ)
    Delay(80000);   //Waiting for HY3106 ADC convert time
    #endif
    #if defined(HAO_32MHZ)
    Delay(160000);   //Waiting for HY3106 ADC convert time
    #endif
    //HY3106 : ADGN=x1, PGA=x8, FRb=x1/2, VDDA=2.4V.
    //ADC Vin range = +/- 0.15V
    spi_read_buffer[2]=spi_read_24bits(0xA0);
```

```
      ADCData=spi_read_buffer[2];
      LCD_DATA_DISPLAY(ADCData);
   }

}

/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description   : I2C/UART/SPI interrupt Service Routine (HW0).             */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
   //SPI RX interrupt
   while(DrvSPI32_GetRxIntFlag())
   {
      dummyread =    (unsigned char) DrvSPI32_Read();
      while((spi_00>>BUF) & 0x01);    // Wait SPI Busy
      DrvSPI32_ClrIntRxFlag();
      rxIntHappened=1;
   }
}
/*---------------------------------------------------------------------------*/
/* 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   : LVD & BOR2 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: HW7_ISR()                                                */
/* Description  : UART2 interrupt Service Routine (HW7).                   */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                */
/* Description  : TMB2 interrupt Service Routine (HW8).                    */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                */
/* Description  : PT3 interrupt Service Routine (HW9).                     */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                              */
/* Description  : Exception Service Routines.                              */
/* Arguments    : None.                                                    */
```

```
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                */
/* Description  : Exception Service Routines.                              */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*-------------------------------------------------------------------------*/
/* Software Delay Subroutines                                              */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
   asm("NOP");
}
/*-------------------------------------------------------------------------*/
/* SPI                                                                     */
/*-------------------------------------------------------------------------*/
static void doClearSPIFlag(void)
{
   rxIntHappened=0;
}
static void waitSPIDONE(void)
{
   while(!rxIntHappened)
   {
      #if defined(HAO_4MHZ)
      enterIdleMode();
      #else
      asm("NOP");
      #endif
   }
}

#if defined(SPI_4WIRE)
void spi_write_8bits(uint32_t spi_address, uint32_t spi_data)
{
   DrvSPI32_BitLength(16);       //write MSB 16bits
   SPI_WRITE8=(spi_address<<8) + spi_data;
   doClearSPIFlag();
   DrvSPI32_Write(SPI_WRITE8);
   waitSPIDONE();
}
```

```
void spi_write_16bits(uint32_t spi_address, uint32_t spi_data)
{
   DrvSPI32_BitLength(24);     //write MSB 24bits
   SPI_WRITE16=(spi_address<<16) + spi_data;
   doClearSPIFlag();
   DrvSPI32_Write(SPI_WRITE16);
   waitSPIDONE();
}
unsigned int spi_read_8bits(uint32_t spi_address)
{
   unsigned int au32DestinationData=0;
   DrvSPI32_BitLength(16);     //write MSB 16bits
   doClearSPIFlag();
   DrvSPI32_Write((spi_address<<8) + 0xFF);
   waitSPIDONE();
   au32DestinationData=DrvSPI32_Read();
   return(au32DestinationData&0x000000FF);
}
unsigned int spi_read_16bits(uint32_t spi_address)
{
   unsigned int au32DestinationData=0;
   DrvSPI32_BitLength(24); //write MSB 24bits
   doClearSPIFlag();
   DrvSPI32_Write((spi_address<<16) + 0xFF);
   waitSPIDONE();
   au32DestinationData=DrvSPI32_Read();
   return(au32DestinationData&0x0000FFFF);
}
unsigned int spi_read_24bits(uint32_t spi_address)
{
   unsigned int au32DestinationData=0;
   DrvSPI32_BitLength(32);
   SPI_WRITE32=spi_address<<24;
   doClearSPIFlag();
   DrvSPI32_Write(SPI_WRITE32);
   waitSPIDONE();
   au32DestinationData=DrvSPI32_Read();
   return(au32DestinationData & 0x00FFFFFF );
}
#endif

#if defined(SPI_3WIRE)
void spi_write_8bits(uint32_t spi_address, uint32_t spi_data)
{
   DrvGPIO_ClrBit(E_PT2,SCS_PIN);                    //CS: Low
   DrvSPI32_BitLength(16);
   SPI_WRITE8=(spi_address<<8) + spi_data;
   doClearSPIFlag();
   DrvSPI32_Write(SPI_WRITE8);
   waitSPIDONE();
   DrvGPIO_SetBit(E_PT2,SCS_PIN);                     //CS: High
}
void spi_write_16bits(uint32_t spi_address, uint32_t spi_data)
{
   DrvGPIO_ClrBit(E_PT2,SCS_PIN);                    //CS: Low
   DrvSPI32_BitLength(24);                          //write MSB 24bits
   SPI_WRITE16=(spi_address<<16) + spi_data;
   doClearSPIFlag();
```

```
    DrvSPI32_Write(SPI_WRITE16);
    waitSPIDONE();
    DrvGPIO_SetBit(E_PT2,SCS_PIN);                    //CS: High
}
unsigned int spi_read_8bits(uint32_t spi_address)
{
    DrvGPIO_ClrBit(E_PT2,SCS_PIN);                    //CS: Low
    unsigned int au32DestinationData=0;
    DrvSPI32_BitLength(16);                           //write MSB 16bits
    doClearSPIFlag();
    DrvSPI32_Write((spi_address<<8) + 0xFF);
    waitSPIDONE();
    au32DestinationData=DrvSPI32_Read();
    DrvGPIO_SetBit(E_PT2,SCS_PIN);                    //CS: High
    return(au32DestinationData&0x000000FF);
}
unsigned int spi_read_16bits(uint32_t spi_address)
{
    DrvGPIO_ClrBit(E_PT2,SCS_PIN);                    //CS: Low
    unsigned int au32DestinationData=0;
    DrvSPI32_BitLength(24);                           //write MSB 24bits
    doClearSPIFlag();
    DrvSPI32_Write((spi_address<<16) + 0xFF);
    waitSPIDONE();
    au32DestinationData=DrvSPI32_Read();
    DrvGPIO_SetBit(E_PT2,SCS_PIN);                    //CS: High
    return(au32DestinationData&0x0000FFFF);
}

unsigned int spi_read_24bits(uint32_t spi_address)
{
    DrvGPIO_ClrBit(E_PT2,SCS_PIN);                    //CS: Low
    unsigned int au32DestinationData=0;
    DrvSPI32_BitLength(32);
    SPI_WRITE32=spi_address<<24;
    doClearSPIFlag();
    DrvSPI32_Write(SPI_WRITE32);
    waitSPIDONE();
    au32DestinationData=DrvSPI32_Read();
    DrvGPIO_SetBit(E_PT2,SCS_PIN);                    //CS: High
    return(au32DestinationData & 0x00FFFFFF );
}
#endif
/*---------------------------------------------------------------------------*/
/* POWER                                                                     */
/*---------------------------------------------------------------------------*/
void enterIdleMode(void)
{
    SOCSTATUSL = 0x1010; // IDLE
    asm("STANDBY wake_grant");
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalSPI(void)                                            */
/* Description   :                                                           */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
```

```
/*---------------------------------------------------------------------------*/
void InitalSPI(void)
{
   Initial_SPI32_Register();       //Reset SPI Register
   DrvSPI32_CLKSource(1);
   DrvGPIO_Open(E_PT2,Flash_MOSI|Flash_SCLK,E_IO_OUTPUT);    //MOSI=PT23, SCK=PT21
   DrvGPIO_Open(E_PT2,Flash_MISO,E_IO_INPUT);    //MISO=PT22
   DrvGPIO_Open(E_PT2,Flash_CS,E_IO_OUTPUT);    //CS=PT2.0 output
   DrvGPIO_SetPortBits(E_PT2,Flash_CS); //CS=PT2.0 high
#if defined(SPI_3WIRE)
   DrvSPI32_Open(E_DRVSPI_MASTER2,E_DRVSPI_TYPE1,2,5);
#endif
#if defined(SPI_4WIRE)
   DrvSPI32_Open(E_DRVSPI_MASTER1,E_DRVSPI_TYPE1,2,5);
#endif
   // Master2 : 3-wire mode
   // Master1 : 4-wire mode
   // Type1 : CPHA=0 CPOL=1
   // Type3 : CPHA=1 CPOL=1
   // 2 : PT2.0=CS PT2.1=CK P2.2=MISO PT2.3=MOSI
   // 5 : CLOCK/128
   DrvSPI32_Enable();
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

13. Communication IP(UART)

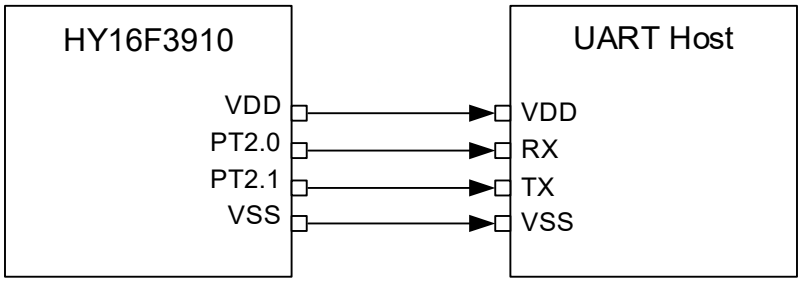
13.1. Example Name

HY16F3910_UART

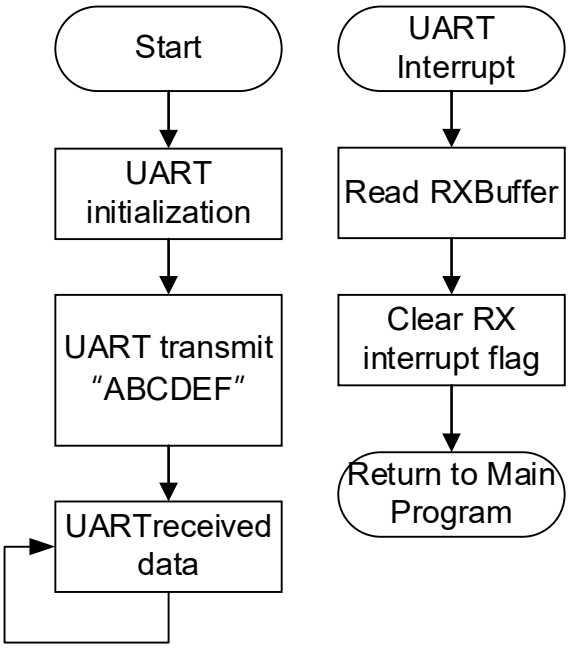
13.2. Example Description

(1) HY16F198B UART TX and RX tutorial.
(2) In this example code, using #define to select UART port is PORT9, otherwise UART port is PT2 port. If UART port is Port2, TX=PT2.0, RX=PT2.1. If UART port is PORT9, TX=PT9.0, RX=PT9.1
(3) using #define HAO_4MHZ/ HAO_32MHZ 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'

13.3. System Description



13.4. Software Flowchart



13.5. 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 : HY16F3910_UART
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2022/03/22
 * Created by    :
 *
 * Program Description   : using #define HAO_4MHZ/HAO_32MHZ to select HAO frequency.
 * 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'
 *
```

```
 *   HY16F3910                          host
 *---------------                 ------------
 *                      |             |
 *                VDD | -----> | VDD
 *                 TX | -----> | RX
 *                 RX | <----- | TX
 *                VSS | -----> | VSS
 *                      |             |
 *---------------                 ------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvUART.h"
#include "DrvPMU.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                                                               */
/*---------------------------------------------------------------------------*/
//PORT
//#define PORT9

//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ

#if defined(PORT9)
#define   UART_PORT E_PT9
#define   UART_TXD   BIT0
#define   UART_RXD   BIT1
```

```
#else
#define   UART_PORT E_PT2
#define   UART_TXD   BIT0
#define   UART_RXD   BIT1
#endif

#define   Uart_RX_BufferSize 6
#define   Uart_TX_BufferSize 8

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char UartRxBuffer[Uart_RX_BufferSize]={0},UartTxBuffer[Uart_TX_BufferSize]={0};
unsigned char UartTxIndex,UartTxLength,UartRxIndex,UartRxLength;
MCUSTATUS   MCUSTATUSbits;

unsigned char UartRxBuffer_Command[Uart_RX_BufferSize]=
{
0x61,0x62,0x63,0x64,0x65,0x66 //ASCII   KEYWORD = abcdef
};
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalUart(void);
void InitalClock(void);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  unsigned char Stop_To_Send_UART=DISABLE;
  InitalClock();
  InitalUart();
  MCUSTATUSbits._byte = 0;
  MCUSTATUSbits.b_UART_TxDone=ENABLE;
  SYS_EnableGIE(4,0x3BF);   //Enable GIE(Global Interrupt)

  UartTxIndex=0;
  UartRxIndex=0;

  while(1)
  {Delay(50000);
    //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)
{
   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)
        {
            while(DrvUART_TRStatus(3));    //wait TX finish!!
            DrvUART_EnableInt(DISABLE,ENABLE);    //DISABLE UART Tx Interrupt, ENABLE UART Rx Interrupt
            DrvUART_ClrTxFlag();
        }
    }

}
/*---------------------------------------------------------------------------*/
/* 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    : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
/* Description  : PT3 interrupt Service Routine (HW9).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{
```

```
}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalClock()                                              */
/* Description   : CLOCK Initial Subroutines.                                */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void InitalClock(void)
{
   DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
        DrvCLOCK_EnableHighOSC(E_INTERNAL,100);       // Select HSRC
      #if defined(HAO_4MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                  //Select internal 4.147MHz, and calibration
4.147MHz
        DrvCLOCK_SelectMCUClock(0,3);                 //ENMCD=01b,MCU Clock/1, CPU CLOCK IS 'HS_CK/1'
      #elif defined(HAO_32MHZ)
        DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                   //Select internal 31.795MHz, and
calibration 31.795MHz
        DrvCLOCK_SelectMCUClock(0,0);                 //ENMCD=00b,MCU Clock/2, CPU CLOCK IS 'HS_CK/2'
      #endif
#endif

#if defined(HSXT)
      DrvCLOCK_SelectOHS_HS(1);    //0:HSXT<4MHZ 1:HSXT>4MHZ
      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)

  #if defined(PORT9)
    #if defined(HAO_4MHZ)
    DrvUART_ClkEnable(1,0);//TUCK=1b(HSRC), ENUD=1b, UACD=000b
    DrvUART_Open(4147,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,6);
    #endif
    #if defined(HAO_32MHZ)
    DrvUART_ClkEnable(1,0);//UACD=000b, EUART/1, 31.795M/1=31.795Mhz

DrvUART_Open(31795,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,6);
  #endif
  #else //PORT2
    #if defined(HAO_4MHZ)
    DrvUART_ClkEnable(1,0);//TUCK=1b(HSRC), ENUD=1b, UACD=000b
    DrvUART_Open(4147,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,2);
    #endif
    #if defined(HAO_32MHZ)
    DrvUART_ClkEnable(1,0);//UACD=000b, EUART/1, 31.795M/1=31.795Mhz

DrvUART_Open(31795,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,2);
    #endif
  #endif
#endif

  //4147 : oscillator frequency 4MHz Unit After Calibration HAO = 4147kHz
  //31795 : oscillator frequency 32MHz Unit After Calibration HAO = 31795kHz
  //None parity
  //8 data bits.
  //6 : Port 9.0 =TX, Port 9.1 =RX
  //2 : Port 2.0 =TX, Port 2.1 =RX

#if defined(HSXT)//External 16MHz
  #if defined(PORT9)
  DrvUART_ClkEnable(0,0);          //TUCK=0b(HSXT), ENUD=1b, UACD=000b, select External 16MHz
  DrvUART_Open(16000,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,6);
  #else //PORT2
  DrvUART_ClkEnable(0,0);          //TUCK=0b(HSXT), ENUD=1b, UACD=000b, select External 16MHz
  DrvUART_Open(16000,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,2);
  #endif
#endif

#if defined(PORT9)
  DrvGPIO_LCDIOOpen(UART_PORT, UART_TXD, E_IO_OUTPUT);
  DrvGPIO_LCDIOOpen(UART_PORT, UART_RXD, E_IO_INPUT);
#else
  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);
#endif

    DrvUART_EnableInt(ENABLE,ENABLE); //Enable UART Tx Interrupt, Enable UART Rx Interrupt
    DrvUART_Disable_AutoBaudrate();
    DrvUART_Close();
    DrvUART_Enable();
}
/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for(;num>0;num--)
    asm("NOP");
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

14. Communication IP(UART2)

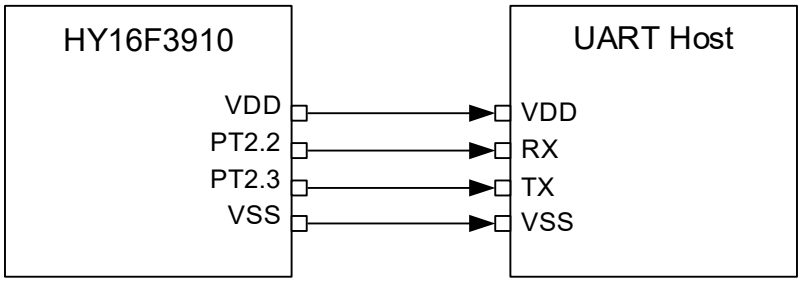
14.1. Example Name

HY16F3910_UART2

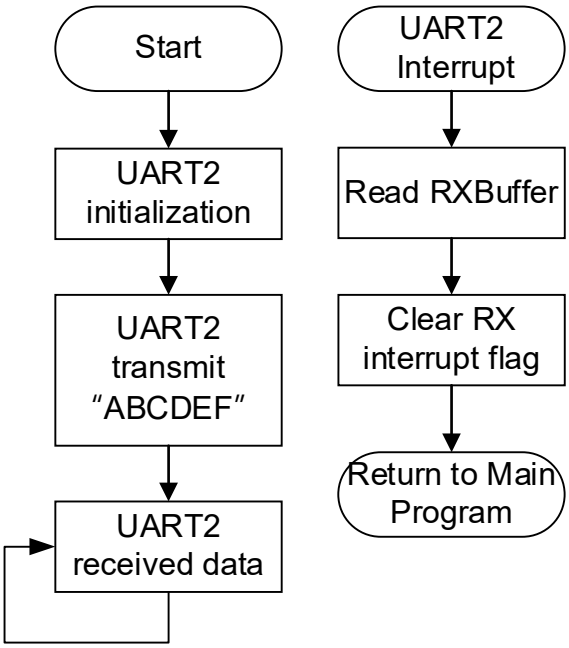
14.2. Example Description

(1) HY16F198B UART2 TX and RX tutorial.
(2) In this example code, using #define to select UART2 port is PORT9, otherwise UART2 port is PT2 port. If UART port is Port2, TX=PT2.2, RX=PT2.3. If UART port is PORT9, TX=PT9.6, RX=PT9.7
(3) using #define HAO_4MHZ/ HAO_32MHZ to select HAO frequency.
(4) UART2 TX continue transmit 'ABCEDF' until RX receiving 'abcedf' to stop transmit. If RX receiving not equal to 'abcedf', UART2 TX start to transmit 'ABCEDF'

14.3. System Description



14.4. Software Flowchart



14.5. 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 : HY16F3910_UART2
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2022/03/22
 * Created by    :
 *
 * Program Description   : using #define HAO_4MHZ/HAO_32MHZ to select HAO frequency.
 * 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'
 *
```

```
 *   HY16F3910                          host
 *--------------                 ------------
 *                      |             |
 *                VDD | -----> | VDD
 *                 TX | -----> | RX
 *                 RX | <----- | TX
 *                VSS | -----> | VSS
 *                      |             |
 *--------------                 ------------
 *
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvUART.h"
#include "DrvPMU.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_UART2_TxDone:1;
      unsigned b_UART2_RxDone:1;
   };
} MCUSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//PORT
//#define PORT9

//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ

#if defined(PORT9)
#define   UART2_PORT E_PT9
#define   UART2_TXD   BIT2
#define   UART2_RXD   BIT3
```

```
#else
#define   UART2_PORT E_PT2
#define   UART2_TXD   BIT2
#define   UART2_RXD   BIT3
#endif

#define   Uart2_RX_BufferSize 6
#define   Uart2_TX_BufferSize 8

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char Uart2RxBuffer[Uart2_RX_BufferSize]={0},Uart2TxBuffer[Uart2_TX_BufferSize]={0};
unsigned char Uart2TxIndex,Uart2TxLength,Uart2RxIndex,Uart2RxLength;
MCUSTATUS   MCUSTATUSbits;

unsigned char Uart2RxBuffer_Command[Uart2_RX_BufferSize]=
{
0x61,0x62,0x63,0x64,0x65,0x66 //ASCII   KEYWORD = abcdef
};
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalUart2(void);
void InitalClock(void);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  unsigned char Stop_To_Send_UART2=DISABLE;
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
  InitalClock();
  InitalUart2();
  MCUSTATUSbits._byte = 0;
  MCUSTATUSbits.b_UART2_TxDone=ENABLE;
  SYS_EnableGIE(4,0x3BF);   //Enable GIE(Global Interrupt)

  Uart2TxIndex=0;
  Uart2RxIndex=0;

  while(1)
  { Delay(50000);
    //UART2 RX
    if(MCUSTATUSbits.b_UART2_RxDone==ENABLE)
    {

      if(
        (Uart2RxBuffer[5]==Uart2RxBuffer_Command[5] && Uart2RxBuffer[4]==Uart2RxBuffer_Command[4]) &&
        (Uart2RxBuffer[3]==Uart2RxBuffer_Command[3] && Uart2RxBuffer[2]==Uart2RxBuffer_Command[2]) &&
        (Uart2RxBuffer[1]==Uart2RxBuffer_Command[1] && Uart2RxBuffer[0]==Uart2RxBuffer_Command[0])
        )
    {
        //if UART2 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_UART2=ENABLE;   //UART stop to send out ABCDEF
        for(Uart2TxLength=0;Uart2TxLength<=Uart2_TX_BufferSize;Uart2TxLength++)
```

```
            {
               Uart2TxBuffer[Uart2TxLength]=Uart2RxBuffer[Uart2TxLength];
               if(Uart2TxLength==Uart2_RX_BufferSize)
               {
                  Uart2RxIndex=0;
                  MCUSTATUSbits.b_UART2_TxDone=DISABLE;
                  DrvUART2_EnableInt(ENABLE,DISABLE);     //Enable UART2 Tx Interrupt, Disable UART2 Rx Interrup
                  while(!MCUSTATUSbits.b_UART2_TxDone); //If    MCUSTATUSbits.b_UART2_TxDone=DISABLE, stop at
here
               }
            }
      }
      else
      {
         //if UART2 receive != 0x61(a)0x62(b)0x63(c)0x64(d)0x65(e)0x66(f)
         //User defined at here
         Stop_To_Send_UART2=DISABLE;     //UART2 start to send out ABCDEF
      }
         Uart2RxIndex=0; //When finished the data reception. Set Uart2RxIndex=0
         MCUSTATUSbits.b_UART2_RxDone=DISABLE;
      }

      //UART2 TX
      if(MCUSTATUSbits.b_UART2_TxDone==ENABLE && Stop_To_Send_UART2==DISABLE )
      {
         Uart2TxBuffer[7]='\r';
         Uart2TxBuffer[6]='\n';
         Uart2TxBuffer[5]=0x46;   //F
         Uart2TxBuffer[4]=0x45;   //E
         Uart2TxBuffer[3]=0x44;   //D
         Uart2TxBuffer[2]=0x43;   //C
         Uart2TxBuffer[1]=0x42;   //B
         Uart2TxBuffer[0]=0x41;   //A
         MCUSTATUSbits.b_UART2_TxDone=DISABLE;
         Uart2TxLength=8;
         Uart2TxIndex=0;
         DrvUART2_EnableInt(ENABLE,ENABLE);        //Enable UART2 Tx Interrupt;Enable UART2 Rx Interrupt
         while(!MCUSTATUSbits.b_UART2_TxDone);   //If    MCUSTATUSbits.b_UART2_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)
{

}
/*---------------------------------------------------------------------------*/
/* 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   : LVD & BOR2 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: HW7_ISR()                                                */
/* Description   : UART2 interrupt Service Routine (HW7).                  */
/* Arguments      : None.                                                  */
```

```
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW7_ISR(void)
{
  if(DrvUART2_GetRxFlag())
  {
    Uart2RxBuffer[Uart2RxIndex]=DrvUART2_Read();
    Uart2RxIndex++;
    if(Uart2RxIndex>=Uart2_RX_BufferSize)
    {
      Uart2RxIndex=0;
      MCUSTATUSbits.b_UART2_RxDone=ENABLE;
    }
    DrvUART2_ClrRxFlag();
  }

  if(DrvUART2_GetTxFlag())
  {
    if(MCUSTATUSbits.b_UART2_TxDone==DISABLE)
    {
      DrvUART2_Write(Uart2TxBuffer[Uart2TxIndex++]);
      DrvUART2_ClrTxFlag();
      if(Uart2TxIndex>=Uart2TxLength)
      {
        DrvUART2_EnableInt(ENABLE,ENABLE);   //ENABLE UART2 Tx Interrupt, ENABLE UART2 Rx Interrupt
        MCUSTATUSbits.b_UART2_TxDone=ENABLE;
        Uart2TxIndex=0;
      }
    }
    if(MCUSTATUSbits.b_UART2_TxDone==ENABLE)
    {
      while(DrvUART2_TRStatus(3));   //wait TX finish!!
      DrvUART2_EnableInt(DISABLE,ENABLE);   //DISABLE UART2 Tx Interrupt, ENABLE UART2 Rx Interrupt
      DrvUART2_ClrTxFlag();
    }
  }
}
/*-------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                */
/* Description  : TMB2 interrupt Service Routine (HW8).                    */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*-------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                */
/* Description  : PT3 interrupt Service Routine (HW9).                     */
/* Arguments    : None.                                                    */
/* Return Value : None.                                                    */
/* Remark       :                                                          */
/*-------------------------------------------------------------------------*/
void HW9_ISR(void)
{
```

```
}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                                */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
  asm("nop"); //procedure define by customer.
  asm("nop");
  while(1);
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description   : Exception Service Routines.                               */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
  asm("nop"); //procedure define by customer.
  asm("nop");
  while(1);
}

/*---------------------------------------------------------------------------*/
/* Function Name: InitalClock()                                              */
/* Description   : CLOCK Initial Subroutines.                                */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void InitalClock(void)
{
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
      DrvCLOCK_EnableHighOSC(E_INTERNAL,100);       // Select HSRC
    #if defined(HAO_4MHZ)
      DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                  //Select internal 4.147MHz, and calibration
4.147MHz
      DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);              //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
    #elif defined(HAO_32MHZ)
      DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                 //Select internal 31.795MHz, and
calibration 31.795MHz
      DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);              //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
    #endif
#endif
#if defined(HSXT)
  DrvCLOCK_SelectOHS_HS(1);     //0:HSXT<4MHZ 1:HSXT>4MHZ
  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif
}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: InitalUart2()                                              */
/* Description    : UART2 Initial Subroutines.                               */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void InitalUart2(void)
{

#if defined(HSRC)

  #if defined(PORT9)
    #if defined(HAO_4MHZ)
    DrvUART2_ClkEnable(1,0);//TU2CK=1b(HSRC), ENU2D=1b, UACD=000b
      DrvUART2_Open(4147,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,
6);
    #endif
    #if defined(HAO_32MHZ)
    DrvUART2_ClkEnable(1,0);//UACD=011b, EUART/8, 31.795M/8=3.974375MHz

DrvUART2_Open(31795,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,6);
  #endif
  #else //PORT2
    #if defined(HAO_4MHZ)
    DrvUART2_ClkEnable(1,0);//TU2CK=1b(HSRC), ENU2D=1b, UACD=000b
    DrvUART2_Open(4147,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,2);
    #endif
    #if defined(HAO_32MHZ)
    DrvUART2_ClkEnable(1,0);//UACD=011b, EUART/8, 31.795M/8=3.974375MHz

DrvUART2_Open(31795,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,2);
    #endif
  #endif
#endif

  //4147 : oscillator frequency 4MHz Unit After Calibration HAO = 4147kHz
  //31795 : oscillator frequency 16MHz Unit After Calibration HAO = 31795kHz
  //None parity
  //8 data bits.
  //6 : Port 9.2 =TX, Port 9.3 =RX
  //2 : Port 2.2 =TX, Port 2.3 =RX

#if defined(HSXT)
  #if defined(PORT9)
  DrvUART2_ClkEnable(0,0);          //TU2CK=0b(HSXT), ENU2D=1b, UACD=000b, select External 16MHz
  DrvUART2_Open(16000,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,6);
  #else //PORT2
  DrvUART2_ClkEnable(0,0);          //TU2CK=0b(HSXT), ENU2D=1b, UACD=000b, select External 16MHz
  DrvUART2_Open(16000,B115200,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,DRVUART_STOPBITS_1,2);
  #endif
#endif

#if defined(PORT9)
  DrvGPIO_LCDIOOpen(UART2_PORT, UART2_TXD, E_IO_OUTPUT);
  DrvGPIO_LCDIOOpen(UART2_PORT, UART2_RXD, E_IO_INPUT);
#else
```

```
    DrvGPIO_Open(UART2_PORT,UART2_TXD,E_IO_OUTPUT);
    DrvGPIO_Open(UART2_PORT,UART2_RXD,E_IO_INPUT);
    DrvGPIO_Open(UART2_PORT,UART2_RXD|UART2_TXD,E_IO_PullHigh);
#endif

    DrvUART2_EnableInt(ENABLE,ENABLE); //Enable UART2 Tx Interrupt, Enable UART2 Rx Interrupt
    DrvUART2_Disable_AutoBaudrate();
    DrvUART2_Close();
    DrvUART2_Enable();
}


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

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

15. Communication IP(I2C)

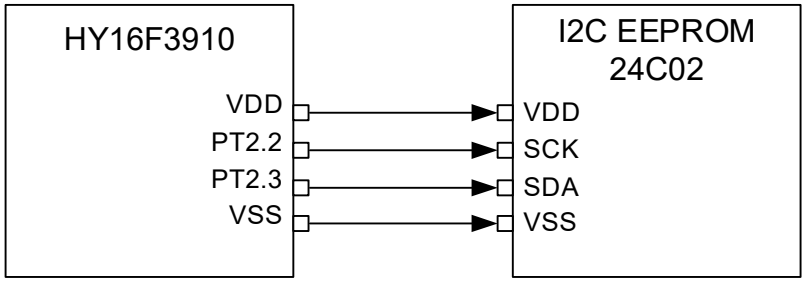
15.1. Example Name

HY16F3910_I2C

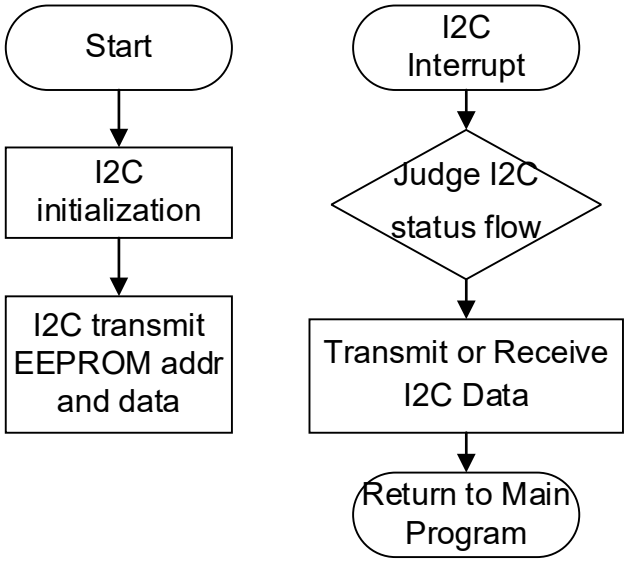
15.2. Example Description

(1) HY16F3910 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.

15.3. System Description



15.4. Software Flowchart



15.5. 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 : HY16F3910_I2C
 * IDE tooling   : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.   :
 * Library Ver. : 0.5
 * MCU Device    :
 * Description   :
 * Created Date : 2022/03/22
 * Created by    : Optimize the I2C interrupt process -Cyril

 * Program Description:
 * --------------------
 * HY16F3910
 * ---------
 *              |                    --------
 *      PT2.3 | SDA <--> SDA |EEPROM|
 *      PT2.2 | SCK ---> SCK --------
```

```
 *              |
 *       GND    |
 *              |
 * ---------
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "main.h"
#include "System.h"
#include "DrvI2C.h"
#include "DrvPMU.h"
#include "DrvClock.h"
#include "EEPROM_24Cxx.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
void I2C_Reset(void);
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//OSC
//#define HSXT    //External 16MHz
#define HSRC    //Internal HAO
//#define HAO_4MHZ
#define HAO_32MHZ


#define   I2C_PORT E_PT2
#define   SCL_PIN 2
#define   SDA_PIN 3
#define   SCL_BIT BIT2
#define   SDA_BIT BIT3
#define   I2CBufferSize   256
#define   I2C_WRITE   0x00          // I2C WRITE command
#define   I2C_READ    0x01          // I2C READ command

#define   I2C_Delay   500   //I2C_CK=100K @HAO=4MHz 500
/*---------------------------------------------------------------------------*/
/* 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);
void I2CInitial(void);

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

```
/*---------------------------------------------------------------------------*/
int main(void)
{
  unsigned int i;
  unsigned char EEPROM_WriteData[4] = {0x01,0x02,0x03,0x04};
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
#if defined(HSRC)
      DrvCLOCK_EnableHighOSC(E_INTERNAL,100);       // Select HSRC
    #if defined(HAO_4MHZ)
      DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
      DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                    //ENMCD=01b,MCU Clock/1, CPU
CLOCK IS 'HS_CK/1'
    #elif defined(HAO_32MHZ)
      DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                    //Select internal 31.795MHz, and
calibration 31.795MHz
      DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                    //ENMCD=00b,MCU Clock/2, CPU
CLOCK IS 'HS_CK/2'
    #endif
#endif
#if defined(HSXT)
  DrvCLOCK_SelectOHS_HS(1);    //0:HSXT<4MHZ 1:HSXT>4MHZ
  DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);
#endif

  I2CInitial();
  for(i=0;i<=I2CBufferSize;i++)
   {
     I2C_Read_Buffer[i]=0;                    //Initial I2C data buffer=0
   }

  SYS_EnableGIE(4,0x3BF);                    //Enable GIE(Global Interrupt)

  // I2C single I2C_WRITE & I2C_READ
  EEPROM_ByteWrite(0x00,0x00);                    //Single Byte I2C_WRITE word address 0x00, and I2C_WRITE data
0x01
  //Delay(16000);                                 //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(16000);                                 //Delay for EEPROM 24C02 I2C_WRITE Cycle Time 5ms

  // I2C sequential I2C_WRITE & I2C_READ
  EEPROM_WriteArray2(0x01,EEPROM_WriteData,4); //I2C_WRITE array data to the word address from 0x01 to 0x04
  //Delay(16000);                                 //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(16000);                                 //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;
unsigned int Timeout;
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)
            {
              Timeout=0;
              DrvI2C_Ctrl(0,1,0,0);   //I2C as master sends STOP signal
                  while(I2C_Status!=0x30)
                  {
                      I2C_Status=DrvI2C_GetStatusFlag();   // Get I2C Status Flag
                      Timeout++;
                      if(Timeout>=I2C_Delay)
                      {
                          DrvI2C_Ctrl(0,0,0,0);              // Clear all I2C flag
                          break;
                      }
                  }
              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();
        Timeout=0;
        DrvI2C_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
            while(I2C_Status!=0x30)
            {
                I2C_Status=DrvI2C_GetStatusFlag();    // Get I2C Status Flag
                Timeout++;
                if(Timeout>=I2C_Delay)
                {
                    DrvI2C_Ctrl(0,0,0,0);              // Clear all I2C flag
                    break;
                }
            }
        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
   {
       DrvI2C_Reset();     //Reboot I2C and clear interrupt flag
      I2C_EndFlag=1;
   }
}
/*---------------------------------------------------------------------------*/
/* 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   : LVD & BOR2 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: HW7_ISR()                                                  */
/* Description  : UART2 interrupt Service Routine (HW7).                     */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description  : TMB2 interrupt Service Routine (HW8).                      */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                                  */
/* Description  : PT3 interrupt Service Routine (HW9).                       */
/* Arguments    : None.                                                      */
/* Return Value : None.                                                      */
/* Remark       :                                                            */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{

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

```
    asm("nop");
    while(1);
}
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description    : Exception Service Routines.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
    asm("nop"); //procedure define by customer.
    asm("nop");
    while(1);
}

/*---------------------------------------------------------------------------*/
/* I2C Initial Subroutines                                                   */
/*---------------------------------------------------------------------------*/
void I2CInitial(void)
{
#if defined(HSRC)
    DrvI2C_ClkEnable(1);     //I2CCKS=1b=HSRC
#endif
#if defined(HSXT)
    DrvI2C_ClkEnable(0);     //I2CCKS=0b=HSXT
#endif
    //DrvI2C_SlaveSet(0xA0,E_DRVI2C_SLAVE_7BIT,1,1);
    DrvI2C_SetIOPin(5);     //Setting io pin, 5: SCL=PT2.2;SDA=PT2.3
    DrvI2C_Open(255);        //31795000/[4*(255+1)]=31kHz
    //Default CPU clock is 4.147MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 4147000/[4*(99+1)]=10kHz
    DrvI2C_EnableInt(E_DRVI2C_INT_ALL_ENABLE);            // Enable I2C interrupt and error interrupt
}

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

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

16. Peripheral IP(Power)

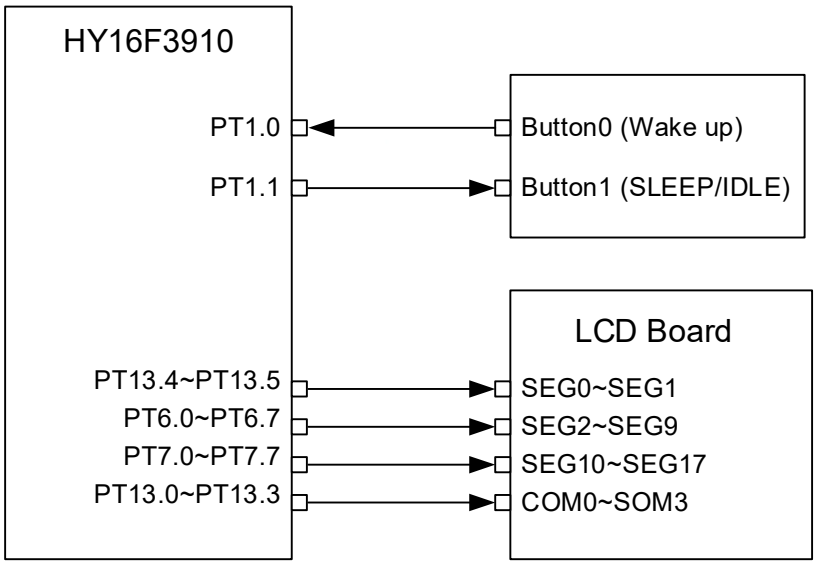
16.1. Example Name

HY16F3910_Power

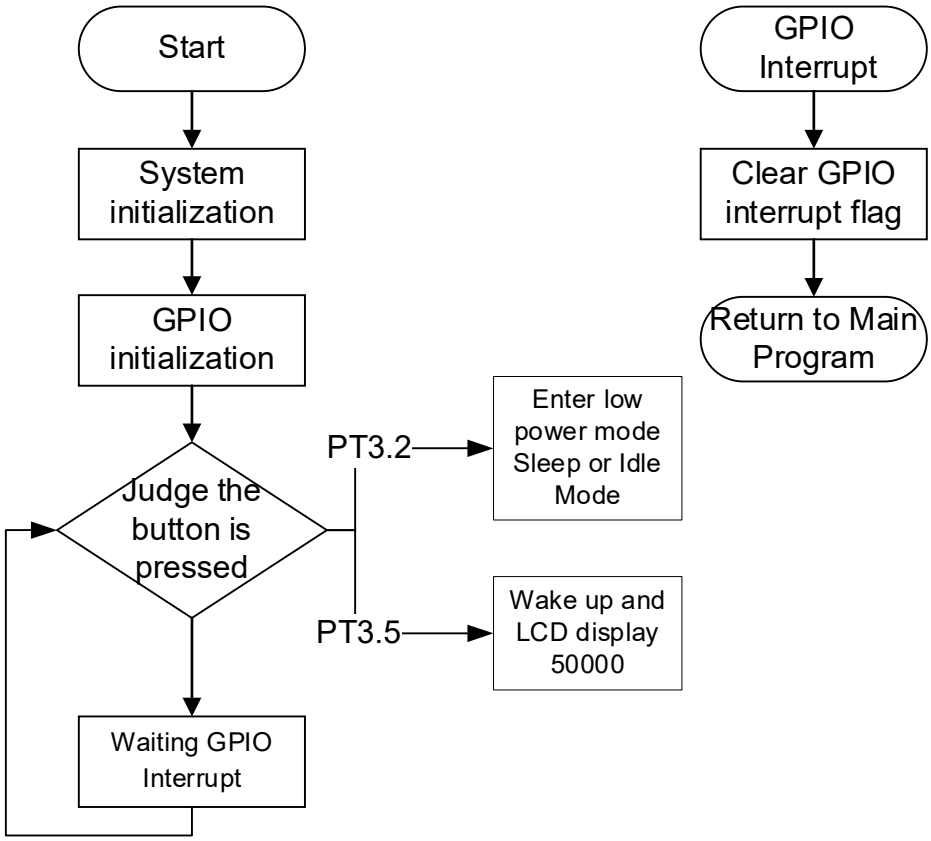
16.2. Example Description

(1) HY16F3910 Sleep and Idle mode tutorial
(2) If press button PT3.2, wake up and LCD display 50000
(3) If press button PT3.5, enter sleep or idle mode. User can select the #define IDLE_MODE or #define SLEEP_MODE to make a decision on sleep or idle mode
(4) If this example code work on HY16F3910 Development board, user can measure Sleep mode current about 1.8uA, Idle mode current about 3.5uA.

16.3. System Description



16.4. Software Flowchart



16.5. 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 : HY16F3910_Power
 * IDE tooling    : AndeSight C/C++ IDE, version: 3.2.1
 * Device Ver.    :
 * Library Ver. : 0.5
 * MCU Device     :
 * Description    :
```

```
 * Created Date : 2022/03/22
 * Created by     :
 *
 * Program Description:
 *
 *    HY16F3910
 *----------------               ----------------
 *                      |           |
 *              PT1.0 | <----- | button0 (Wake up)
 *              PT1.1 | <----- | button1 (IDLE or SLEEP)
 *                      |           |
 *                      |             ----------------
 *                      |
 *                      |               LCD Board(HY10000-AM01)
 *                      |             ----------------
 *                      |           |
 *   PT13.4~PT13.5 | -----> | SEG0~SEG1
 *      PT6.0~PT6.7 | -----> | SEG2~SEG9
 *      PT7.0~PT7.7 | -----> | SEG10~SEG17
 *   PT13.0~PT13.3 | -----> | COM0~COM3
 *                      |           |
 *                      |             ----------------
 *----------------
 *
 * NOTE: Sleep/Idle Current is measured in free run
 ******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY391apb.h"
#include "DrvCLOCK.h"
#include "DrvLCD.h"
#include "Display.h"
#include "main.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvPMU.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
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 BIT0
#define KEYIN1 BIT1

//#define HAO_4MHZ
#define HAO_32MHZ

#define IDLE_MODE
//#define SLEEP_MODE
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
PTINTSTATUS    PT1INTSTATUSbits;

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

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  DrvPMU_VDD15Trim();   // Make sure VDD15 Voltage,before change OSC setting
  DrvCLOCK_EnableHighOSC(E_INTERNAL,100);        // Select HSRC
#if defined(HAO_4MHZ)
  DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_4M);                    //Select internal 4.147MHz, and calibration
4.147MHz
  DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV1);                    //ENMCD=01b,MCU Clock/1, CPU CLOCK IS
'HS_CK/1'
#elif defined(HAO_32MHZ)
  DrvCLOCK_SelectIHOSC_CalHAO(E_HAO_32M);                    //Select internal 31.795MHz, and calibration
31.795MHz
  DrvCLOCK_SelectMCUClock(E_HSCK,MCUCKDIV2);                    //ENMCD=00b,MCU Clock/2, CPU CLOCK IS
'HS_CK/2'
#endif
  DisplayInit();
  ClearLCDframe();
  LCD_DATA_DISPLAY(888888);
  Delay(50000);
  Delay(50000);
  DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_INPUT);          //set PT1.0/PT1.1 INPUT
  DrvGPIO_Open(KEY_PORT,KEYIN1|KEYIN0,E_IO_PullHigh);     //enable PT1.0/PT1.1 pull high R
  DrvGPIO_Open(KEY_PORT,KEYIN0,E_IO_IntEnable);   //PT1.0 interrupt enable
  DrvGPIO_IntTrigger(KEY_PORT,KEYIN0,E_N_Edge);   //PT1.0 interrupt trigger method is negative edge
```

```
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN0);              //clear PT1 interrupt flag
    PT1INTSTATUSbits._byte = 0;
    SYS_EnableGIE(4,0x3BF);                                  //Enable GIE(Global Interrupt)

    while(1)
    {
      DisplayHYcon();
      Delay(10000);
      if((DrvGPIO_PT1_GetPortBits()&KEYIN1)==0)          //if PT1.1 low, Enter Sleep or Idle Mode
      {
        //Close LCD
        DrvLCD_LCDEnable(0);      //LCDEN=0b
        DrvLCD_VLCDEnable(0);     //VLCDEN=0b
        DrvLCD_DisplayMode(2);    //DSP=10b
        DrvLCD_LCDBuffer(0);      //BEn=0b
        while((inw(0x41B00)&(1<<IDF))==0);                //Wait LCD Idle, IDF=20
        //Power
        DrvPMU_DisableBOR2();
        //Enter IDLE or SLEEP
        #if defined(IDLE_MODE)    //Idle Mode
        SYS_LowPower(SYS_IdleMode,0);    //0: HAO Disable 1:HAO Enable
        #endif
        #if defined(SLEEP_MODE)    //Sleep Mode
        DrvPMU_LDO_LowPower(1);     //Low Power mode,only support SLEEP mode
        SYS_LowPower(SYS_SleepMode,0); //0: LPO Disable 1:LPO Enable
        DrvPMU_LDO_LowPower(0);     //SET normal power mode
        #endif
        //If wake up from Sleep or Idle mode, user can enable internal HAO first to do some application
        DrvPMU_EnableBOR2(1);       // Enable BOR2, Suggest to entering "1" for delay about 300us. The delay time is
150us per increases
        DisplayInit();
      }
    }
}

/*---------------------------------------------------------------------------*/
/* 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   : LVD & BOR2 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(E_PT1);
    if((PORT_IntFlag&KEYIN0)==KEYIN0)
    {
       PT1INTSTATUSbits.b_PTINT0done=1;
    }
    DrvGPIO_ClearIntFlag(KEY_PORT,KEYIN0);          //clear PT1.0 interrupt flag
}

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

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

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW7_ISR()                                               */
/* Description    : UART2 interrupt Service Routine (HW7).                */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark            :                                                    */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                               */
/* Description    : TMB2 interrupt Service Routine (HW8).                 */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark            :                                                    */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: HW9_ISR()                                               */
/* Description    : PT3 interrupt Service Routine (HW9).                  */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark            :                                                    */
/*---------------------------------------------------------------------------*/
void HW9_ISR(void)
{

}
/*---------------------------------------------------------------------------*/
/* Function Name: general_exception_handler()                             */
/* Description    : Exception Service Routines.                           */
/* Arguments      : None.                                                 */
/* Return Value : None.                                                   */
/* Remark            :                                                    */
/*---------------------------------------------------------------------------*/
void general_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: debug_exception_handler()                                  */
/* Description   : Exception Service Routines.                               */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void debug_exception_handler()
{
   asm("nop"); //procedure define by customer.
   asm("nop");
   while(1);
}

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

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

17. Revisions

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

Version	Page	Date	Revision Summary
2022/05/18	V04	All	First edition
2022/12/13	V05	All	1. Full sample code link to 16F3910_LibV0.5 2. Chip operating voltage range changed from 1.8V~5.5V to 2.0V~5.5V. 3. Full sample code DrvCLOCK_SelectIHOSC() changed to DrvCLOCK_SelectIHOSC_CalHAO() function. 4. Full sample code modification of high-speed external oscillator start-up process. 5. Fixed RTC interrupt trigger flag for RTC demo code 6. Fixed WDT demo code to clear WDT counter in WDT interrupt