Micro-Controller Experiment

Week6

Teacher: 廖裕評 Yu-Ping Liao

TA: 陳大荃 Da-chuan Chen, 陳恩妮 En-ni Chen

Class Rules

1. No drink besides water.
2. Bring a laptop and breadboard if needed.
3. Ask us TAs to sign and borrow development boards. Do not sign or ask others to sign for you without TAs' permission.
4. Arriving 10 minutes after the bell rings will be regarded as absent.
5. If you damage any borrowed equipment, you have to pay for it.

Homework Rules

1. Includes: A. Class content, B. Class exercise, C. Homework (screenshot or video)
2. Editing software: MS PowerPoint
3. File format: PDF
4. Filename: "date_group_studentID_name.pdf", like "0916_第1組_11028XXX_陳OO.pdf"
5. The homework deadline is 23:59 of the day before the next class. If you are late, then your grade will be deducted.

Contact

If you encounter any problems with this class, please get in touch with us with the following E-mails:

1. Teacher, Prof. Yu-Ping Liao 廖裕評： lyp@cycu.org.tw
2. TA, Da-chuan Chen 陳大荃： dachuan516@gmail.com
3. TA, En-ni Chen 陳恩妮： anna7125867@gmail.com

Or visit 篤信 Lab353 for further questions.

Outline of the Week

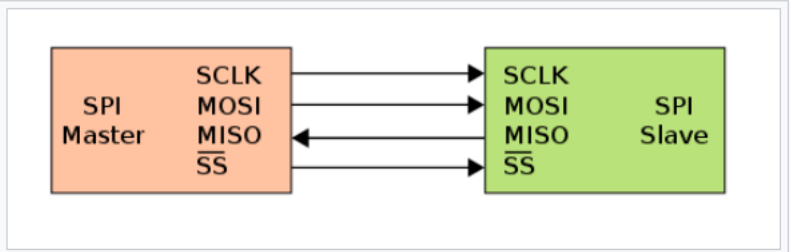
1. SPI introduction
2. SPI Project.
3. Homework 6-1.
4. Homework 6-2.

SPI Introduction

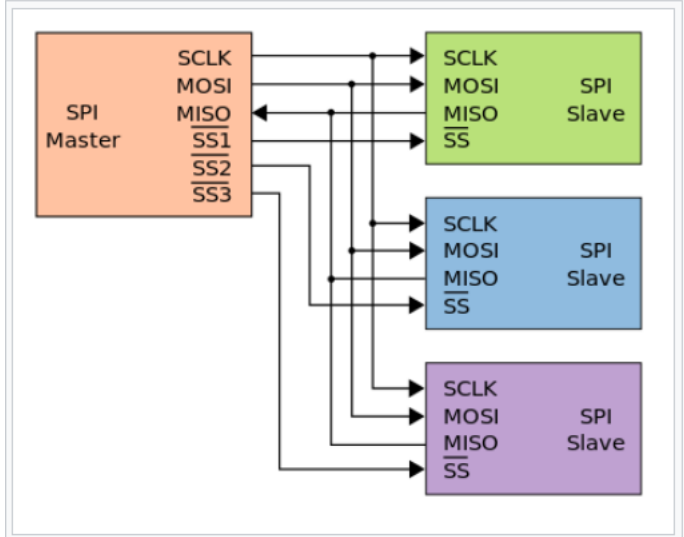
SPI

SPI(Serial Peripheral Interface) is a synchronous serial communication interface specification used for chip communication, primarily employed in single-chip systems. It is similar to I²C.

SPI uses a main–subnode (master/slave) architecture, where one main device orchestrates communication by providing the clock signal and chip select signal(s) which control any number of subservient peripherals.



SPI匯流排：單一主機對單一從機



SPI匯流排：單一主機對複合從機

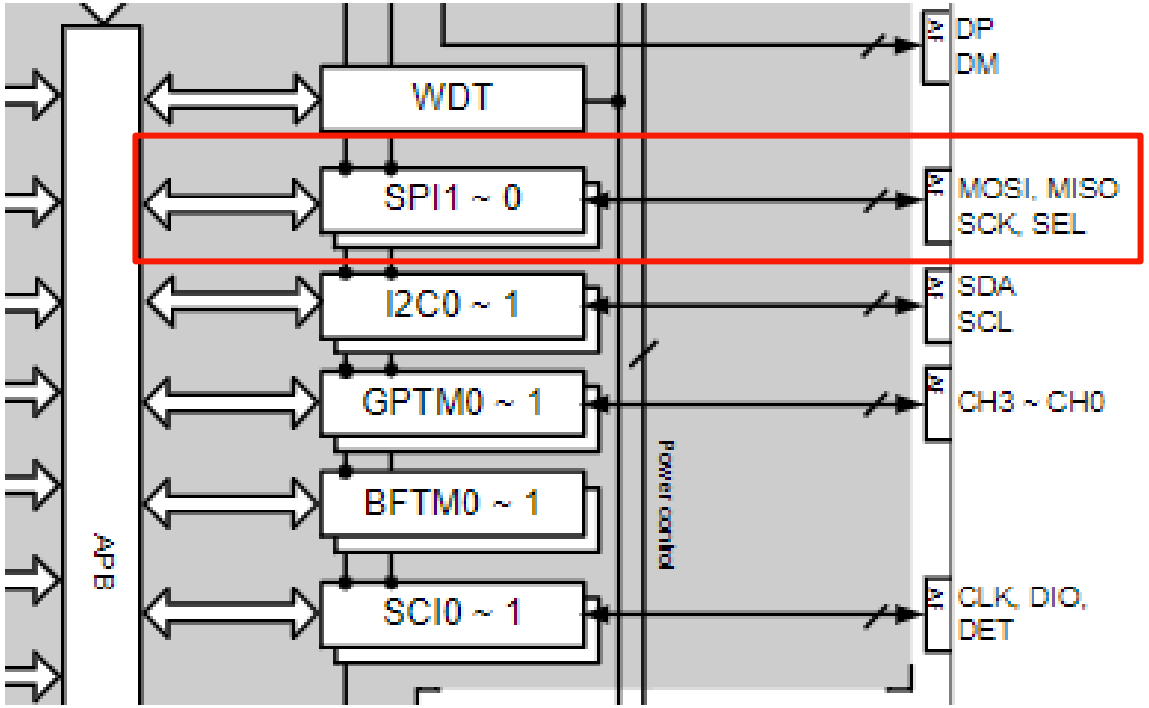
Operation

SCK : Serial Clock (clock signal from main)

MOSI : Main Out Sub In (data output from main)

MISO : Main In Sub Out (data output from sub)

SEL : Slave Select (active low signal from main to address subs and initiate transmission)

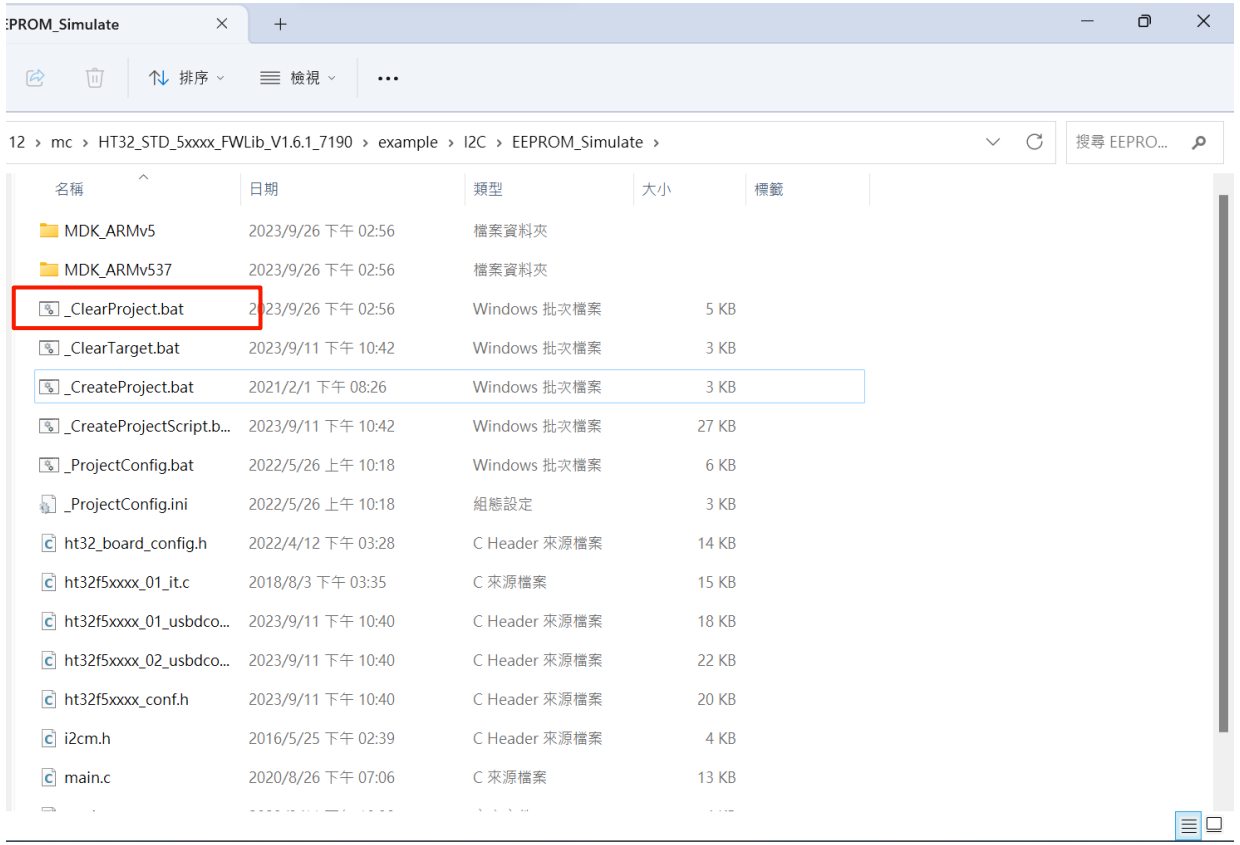


SPI Project

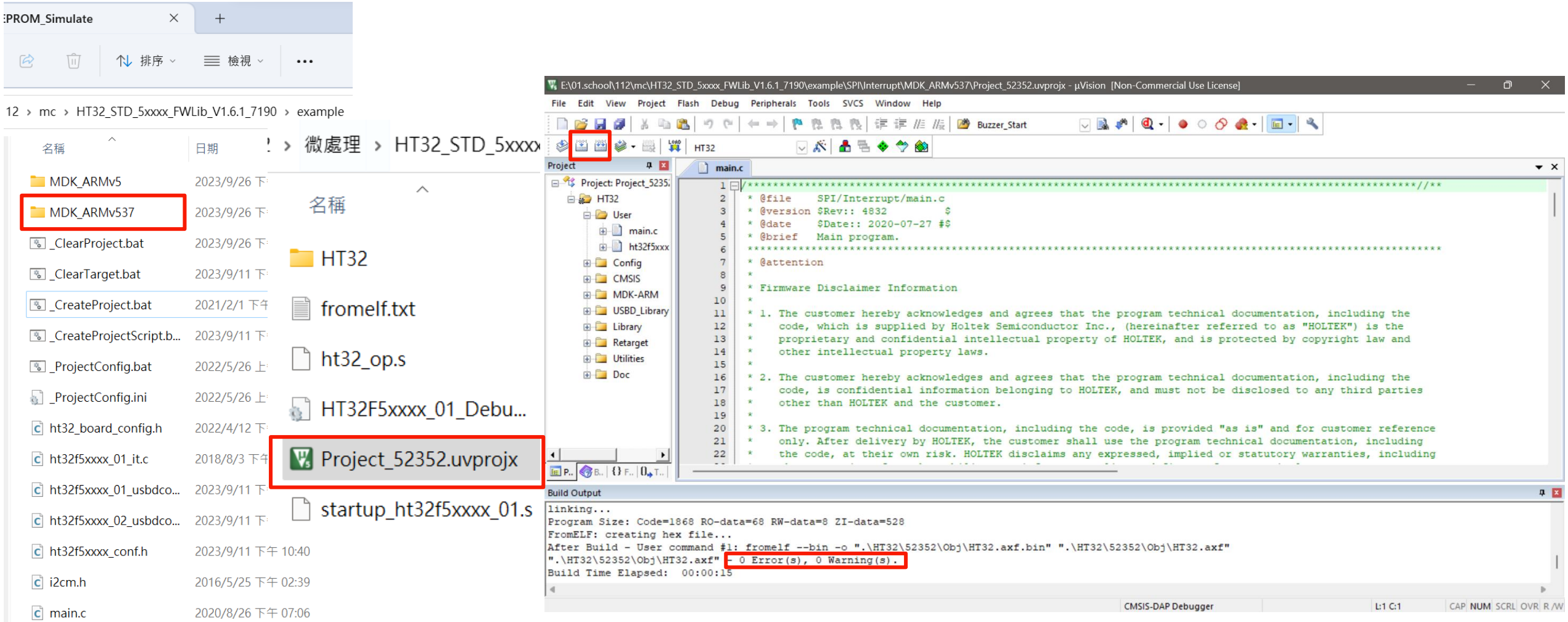
1. Execute “_CreatProject”

1. Go to “~/HT32_STD_5xxxx_FWLib_V1.5.1_7084/example/SPI/Interrupt”.

2. Double click “_CreateProject.bat”.



2. Launch project



main

```
int main(void)
{
  /* Initialize LED1 & LED2 on HT32 board                                    */
  HT32F_DVB_LEDInit(HT_LED1);
  HT32F_DVB_LEDInit(HT_LED2);

  SPI_Configuration();

  SPI_Loopback();

  while (1);
}
```

LED Initialization Setting

SPI Initialization Setting

SPI Return Function Testing

SPI Initialization Setting

```c
void SPI_Configuration(void)
{
  /* !!! NOTICE !!!
     Notice that the local variable (structure) did not have an initial value.
     Please confirm that there are no missing members in the parameter settings below in this function.
  */
  SPI_InitTypeDef SPI_InitStructure;

  CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
  /* Enable Px, Master, Slave  & AFIO clock                                    */
  HTCFG_SPI_MASTER_SEL_GPIO_CLOCK(CKCUClock)          = 1;
  HTCFG_SPI_SLAVE_CLOCK(CKCUClock)                    = 1;
  HTCFG_SPI_MASTER_CLOCK(CKCUClock)                   = 1;
  CKCUClock.Bit.AFIO                                  = 1;
  CKCU_PeripClockConfig(CKCUClock, ENABLE);

  /* MASTER SEL idle state is HIGH                                             */
  GPIO_PullResistorConfig(HTCFG_SPI_MASTER_SEL_GPIO_ID, HTCFG_SPI_MASTER_SEL_AFIO_PIN, GPIO_PR_UP);

  /* Configure related IO to Master mode                                       */
  AFIO_GPxConfig(HTCFG_SPI_MASTER_SEL_AFIO_PORT, HTCFG_SPI_MASTER_SEL_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_MASTER_SCK_AFIO_PORT, HTCFG_SPI_MASTER_SCK_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_MASTER_MOSI_AFIO_PORT, HTCFG_SPI_MASTER_MOSI_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_MASTER_MISO_AFIO_PORT, HTCFG_SPI_MASTER_MISO_AFIO_PIN, AFIO_FUN_SPI);

  /* Configure related IO to Slave mode                                        */
  AFIO_GPxConfig(HTCFG_SPI_SLAVE_SEL_AFIO_PORT, HTCFG_SPI_SLAVE_SEL_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_SLAVE_SCK_AFIO_PORT, HTCFG_SPI_SLAVE_SCK_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_SLAVE_MOSI_AFIO_PORT, HTCFG_SPI_SLAVE_MOSI_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_SLAVE_MISO_AFIO_PORT, HTCFG_SPI_SLAVE_MISO_AFIO_PIN, AFIO_FUN_SPI);

  /* SPI configuration: Master mode                                            */
  SPI_InitStructure.SPI_Mode = SPI_MASTER;
  SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
  SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_8;
  SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
  SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
  SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
  SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
  SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
  SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
  SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
  SPI_InitStructure.SPI_ClockPrescaler = 4;
  SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);

  /* SPI1 configuration: Slave mode                                            */
  SPI_InitStructure.SPI_Mode = SPI_SLAVE;
  SPI_Init(HTCFG_SPI_SLAVE, &SPI_InitStructure);
```

System Clock Setting

Set master SEL as Pullup-Resister

Master Pin Setting

Slave Pin Setting

AFIO Setting

- DataSheet P28

AF0	AF1	AF2	AF3	AF4	AF5
System Default	GPIO	ADC	CMP	MCTM /GPTM	SPI
PA11				MT_ CH1N	SPI0_ MISO
SWCLK	PA12				
SWDIO	PA13				
PA14				MT_ CH0	SPI1_ SEL
PA15				MT_ CH0N	SPI1_ SCK
VDD_2					
VSS_2					
PB0				MT_ CH1	SPI1_ MOSI
PB1				MT_ CH1N	SPI1_ MISO
PD1				MT_ CH2	
PD2				MT_ CH2N	
PD3				MT_ CH3	
VDD_2					
VSS_2					
PB2				MT_ CH2	SPI0_ SEL
PB3				MT_ CH2N	SPI0_ SCK
PB4				MT_ BRK	SPI0_ MOSI
PB5				MT_ BRK	SPI0_ MISO

main.c | ht32_board_config.h

```
86   ******************************************************************************
87   void SPI_Configuration(void)
88   {
89     /* !!! NOTICE !!!
90        Notice that the local variable (structure) did not have an initial value.
91        Please confirm that there are no missing members in the parameter settings below in this fur
92     */
93     SPI_InitTypeDef SPI_InitStructure;
94
95     CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
96     /* Enable Px, Master, Slave  & AFIO clock
97     HTCFG_SPI_MASTER_SEL_GPIO_CLOCK(CKCUClock)          = 1;
98     HTCFG_SPI_SLAVE_CLOCK(CKCUClock)                    = 1;
99     HTCFG_SPI_MASTER_CLOCK(CKCUClock)                   = 1;
100    CKCUClock.Bit.AFIO                                  = 1;
101    CKCU_PeripClockConfig(CKCUClock, ENABLE);
102
103    /* MASTER_SEL idle state is HIGH
104    GPIO_PullResistorConfig(HTCFG_SPI_MASTER_SEL_GPIO_ID, HTCFG_SPI_MASTER_SEL_AFIO_PIN, GPIO_PR_UI
105
106    /* Configure related IO to Master mode
107    AFIO_GPxConfig(HTCFG_SPI_MASTER_SEL_AFIO_PORT, HTCFG_SPI_MASTER_SEL_AFIO_PIN, AFIO_FUN_SPI);
108    AFIO_GPxConfig(HTCFG_SPI_MASTER_SCK_AFIO_PORT, HTCFG_SPI_MASTER_SCK_AFIO_PIN, AFIO_FUN_SPI);
109    AFIO_GPxConfig(HTCFG_SPI_MASTER_MOSI_AFIO_PORT, HTCFG_SPI_MASTER_MOSI_AFIO_PIN, AFIO_FUN_SPI);
110    AFIO_GPxConfig(HTCFG_SPI_MASTER_MISO_AFIO_PORT, HTCFG_SPI_MASTER_MISO_AFIO_PIN, AFIO_FUN_SPI);
```

F 12

```
137  #define HTCFG_SPI_MASTER_CLOCK(CK)                (CK.Bit.SPI0)
138  #define HTCFG_SPI_MASTER                          (HT_SPI0)
139  #define HTCFG_SPI_MASTER_IRQn                     (SPI0_IRQn)
140  #define HTCFG_SPI_MASTER_SEL_AFIO_PORT            (GPIO_PB)
141  #define HTCFG_SPI_MASTER_SCK_AFIO_PORT            (GPIO_PB)
142  #define HTCFG_SPI_MASTER_MOSI_AFIO_PORT           (GPIO_PB)
143  #define HTCFG_SPI_MASTER_MISO_AFIO_PORT           (GPIO_PB)
144  #define HTCFG_SPI_MASTER_SEL_AFIO_PIN             (AFIO_PIN_2)
145  #define HTCFG_SPI_MASTER_SCK_AFIO_PIN             (AFIO_PIN_3)
146  #define HTCFG_SPI_MASTER_MOSI_AFIO_PIN            (AFIO_PIN_4)
147  #define HTCFG_SPI_MASTER_MISO_AFIO_PIN            (AFIO_PIN_5)
148  #define HTCFG_SPI_MASTER_IRQHandler               (SPI0_IRQHandler)
149
150  #define HTCFG_SPI_SLAVE_CLOCK(CK)                 (CK.Bit.SPI1)
151  #define HTCFG_SPI_SLAVE                           (HT_SPI1)
152  #define HTCFG_SPI_SLAVE_IRQn                      (SPI1_IRQn)
153  #define HTCFG_SPI_SLAVE_SEL_AFIO_PORT             (GPIO_PA)
154  #define HTCFG_SPI_SLAVE_SCK_AFIO_PORT             (GPIO_PC)
155  #define HTCFG_SPI_SLAVE_MOSI_AFIO_PORT            (GPIO_PC)
156  #define HTCFG_SPI_SLAVE_MISO_AFIO_PORT            (GPIO_PC)
157  #define HTCFG_SPI_SLAVE_SEL_AFIO_PIN              (AFIO_PIN_14)
158  #define HTCFG_SPI_SLAVE_SCK_AFIO_PIN              (AFIO_PIN_5)
159  #define HTCFG_SPI_SLAVE_MOSI_AFIO_PIN             (AFIO_PIN_8)
160  #define HTCFG_SPI_SLAVE_MISO_AFIO_PIN             (AFIO_PIN_9)
161  #define HTCFG_SPI_SLAVE_IRQHandler                (SPI1_IRQHandler)
```

SPI Initialization Setting

```c
/* SPI configuration: Master mode
SPI_InitStructure.SPI_Mode = SPI_MASTER;
SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_8;
SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
SPI_InitStructure.SPI_ClockPrescaler = 4;
SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);
```

F12

```c
void SPI_Init(HT_SPI_TypeDef* SPIx, SPI_InitTypeDef* SPI_InitStruct)
{
  u32 tmp;

  /* Check the parameters                                                    */
  Assert_Param(IS_SPI(SPIx));
  Assert_Param(IS_SPI_MODE(SPI_InitStruct->SPI_Mode));
  Assert_Param(IS_SPI_FIFO_SET(SPI_InitStruct->SPI_FIFO));
  Assert_Param(IS_SPI_DATALENGTH(SPI_InitStruct->SPI_DataLength));
  Assert_Param(IS_SPI_SEL_MODE(SPI_InitStruct->SPI_SELMode));
  Assert_Param(IS_SPI_SEL_POLARITY(SPI_InitStruct->SPI_SELPolarity));
  Assert_Param(IS_SPI_CPOL(SPI_InitStruct->SPI_CPOL));
  Assert_Param(IS_SPI_CPHA(SPI_InitStruct->SPI_CPHA));
  Assert_Param(IS_SPI_FIRST_BIT(SPI_InitStruct->SPI_FirstBit));
  Assert_Param(IS_SPI_FIFO_LEVEL(SPI_InitStruct->SPI_RxFIFOTriggerLevel));
  Assert_Param(IS_SPI_FIFO_LEVEL(SPI_InitStruct->SPI_TxFIFOTriggerLevel));
  Assert_Param(IS_SPI_CLOCK_PRESCALER(SPI_InitStruct->SPI_ClockPrescaler));

  /*---------------------------- SPIx Control Register 2 Configuration -------------------------------------*/
  tmp =  SPI_InitStruct->SPI_CPOL;
  if (tmp == SPI_CPOL_LOW)
  {
    tmp |= (0x100 << SPI_InitStruct->SPI_CPHA);
  }
  else
  {
    tmp |= (0x200 >> SPI_InitStruct->SPI_CPHA);
  }

  SPIx->CR1 = SPI_InitStruct->SPI_Mode |  SPI_InitStruct->SPI_DataLength |
              SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
              SPI_InitStruct->SPI_FirstBit | tmp;

  /*---------------------------- SPIx FIFO Control Register Configuration ----------------------------------*/
  SPIx->FCR = SPI_InitStruct->SPI_FIFO | SPI_InitStruct->SPI_TxFIFOTriggerLevel |
             (SPI_InitStruct->SPI_RxFIFOTriggerLevel << 4);

  /*---------------------------- SPIx Clock Prescaler Register Configuration -------------------------------*/
  #if (LIBCFG_SPI_CLK_PRE_V01)
  SPIx->CPR = (SPI_InitStruct->SPI_ClockPrescaler - 1);
  #else
  SPIx->CPR = (SPI_InitStruct->SPI_ClockPrescaler / 2) - 1;
  #endif
}
```

SPICR1(SPI Control Register 1)

SPIFCR(SPI FIFO Control Register)

SPICPR(SPI Clock Prescaler Register)

SPI Control Register 1

- UserManual P488

SPI Control Register 1 – SPICR1

This register specifies the SPI parameters including the data length, the transfer format, the SEL active polarity/ mode, the LSB/MSB control, and the master/slave mode.

Offset: 0x004

Reset value: 0x0000_0000

	31	30	29	28	27	26	25	24
	Reserved							
Type/Reset								

	23	22	21	20	19	18	17	16
	Reserved							
Type/Reset								

	15	14	13	12	11	10	9	8
	Reserved	MODE	SELM	FIRSTBIT	SELAP	FORMAT		
Type/Reset		RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0

	7	6	5	4	3	2	1	0
	Reserved				DFL			
Type/Reset					RW 0	RW 0	RW 0	RW 0

SPI FIFO Control Register

- UserManual P495

SPI FIFO Control Register – SPIFCR

This register contains the related SPI FIFO control including the FIFO enable control and the FIFO trigger level selections.

Offset: 0x018

Reset value: 0x0000_0000

	31	30	29	28	27	26	25	24
	Reserved							
Type/Reset								

	23	22	21	20	19	18	17	16
	Reserved							
Type/Reset								

	15	14	13	12	11	10	9	8
	Reserved					FIFOEN	Reserved	
Type/Reset						RW 0		

	7	6	5	4	3	2	1	0
	RXFTLS				TXFTLS			
Type/Reset	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0

SPI Clock Prescaler Register

- UserManual P491

SPI Clock Prescaler Register – SPICPR

This register specifies the SPI clock prescaler ratio.

Offset: 0x00C

Reset value: 0x0000_0000

	31	30	29	28	27	26	25	24
	Reserved							
Type/Reset								
	23	22	21	20	19	18	17	16
	Reserved							
Type/Reset								
	15	14	13	12	11	10	9	8
	CP							
Type/Reset	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0
	7	6	5	4	3	2	1	0
	CP							
Type/Reset	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0

Calculate the value of register

- Take SPICR1 as an example

Configure parameters:

0X0000 4000 (SPI_MODE)

0X0000 2000 (SPI_SELMODE)

0X0000 0000 (SPI_FIRSTBIT)

0X0000 0000 (SPI_SELPOLARITY)

0X0000 0000 (SPI_CPHA)

0X0000 0000 (SPI_CPOL)

0X0000 0008 (SPI_DATALENGTH)

Calculate according to the fig1 :

0X0000 4000 (SPI_MODE)
0X0000 2000 (SPI_SELMODE)
0X0000 0000 (SPI_FIRSTBIT)
0X0000 0000 (SPI_SELPOLARITY)
0X0000 0000 (SPI_CPHA)
0X0000 0000 (SPI_CPOL)
0X0000 0008 (SPI_DATALENGTH)
+)_____
0X0000 6108
0b0000 0000 0000 0000 01100001 0000 1000

- Fig1:

```
/*---------------------------- SPIx Control Register 2 Configuration ---------
tmp =  SPI_InitStruct->SPI_CPOL;
if (tmp == SPI_CPOL_LOW)
{
  tmp |= (0x100 << SPI_InitStruct->SPI_CPHA);
}
else
{
  tmp |= (0x200 >> SPI_InitStruct->SPI_CPHA);
}

SPIx->CR1 = SPI_InitStruct->SPI_Mode |  SPI_InitStruct->SPI_DataLength |
            SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
            SPI_InitStruct->SPI_FirstBit | tmp;
```

Print on the Tera Term to confirm calculation result

```c
int main(void)
{
  RETARGET_Configuration();
  /* Initialize LED1 & LED2 on HT32 board
  HT32F_DVB_LEDInit(HT_LED1);
  HT32F_DVB_LEDInit(HT_LED2);

  SPI_Configuration();

  SPI_Loopback();

  while (1);
}
```

```c
  /*---------------------------- SPIx Control Register 2 Cor
  tmp =  SPI_InitStruct->SPI_CPOL;
  if (tmp == SPI_CPOL_LOW)
  {
    tmp |= (0x100 << SPI_InitStruct->SPI_CPHA);
  }
  else
  {
    tmp |= (0x200 >> SPI_InitStruct->SPI_CPHA);
  }

  SPIx->CR1 = SPI_InitStruct->SPI_Mode |  SPI_InitStruct->SPI_DataLength |
              SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
              SPI_InitStruct->SPI_FirstBit | tmp;
  printf("CR1: %10x\r\n",SPI_InitStruct->SPI_Mode |  SPI_InitStruct->SPI_DataLength |
              SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
              SPI_InitStruct->SPI_FirstBit | tmp);
```

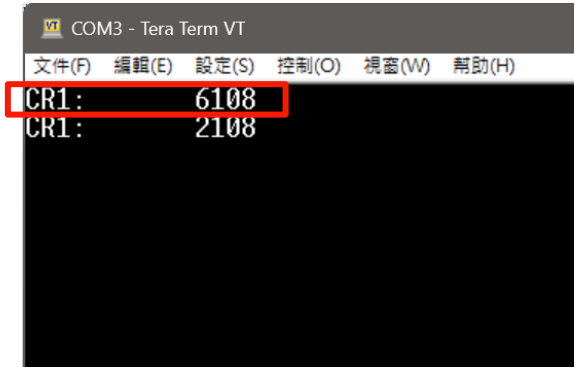
SPI Control Register 1 – SPICR1

This register specifies the SPI parameters including the data length, the transfer format, the SEL active polarity/mode, the LSB/MSB control, and the master/slave mode.

Offset: 0x004

Reset value: 0x0000_0000

	31	30	29	28	27	26	25	24
	Reserved							
Type/Reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
	Reserved							
Type/Reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
	Reserved	MODE	SELM	FIRSTBIT	SELAP	FORMAT		
Type/Reset	0	1	1	0	0	0	0	1
	7	6	5	4	3	2	1	0
	Reserved				DFL			
Type/Reset	RW 0	RW 0	RW 0	RW 0	RW 1	RW 0	RW 0	RW 0



SPI Data Transfer Format

- UserManual P489:

SPI Data Transfer Format

These three bits are used to determine the data transfer format of the SPI interface

FORMAT [2:0]	CPOL	CPHA
001	0	0
010	0	1
110	1	0
101	1	1
Others	Reserved	

CPOL: Clock Polarity

0: SCK Idle state is low

1: SCK Idle state is high

CPHA: Clock Phase

0: Data is captured on the first SCK clock edge

1: Data is captured on the second SCK clock edge

```
/* SPI configuration: Master mode
SPI_InitStructure.SPI_Mode = SPI_MASTER;
SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_8;
SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
SPI_InitStructure.SPI_ClockPrescaler = 4;
SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);
```

CPOL = 0
CPHA = 0

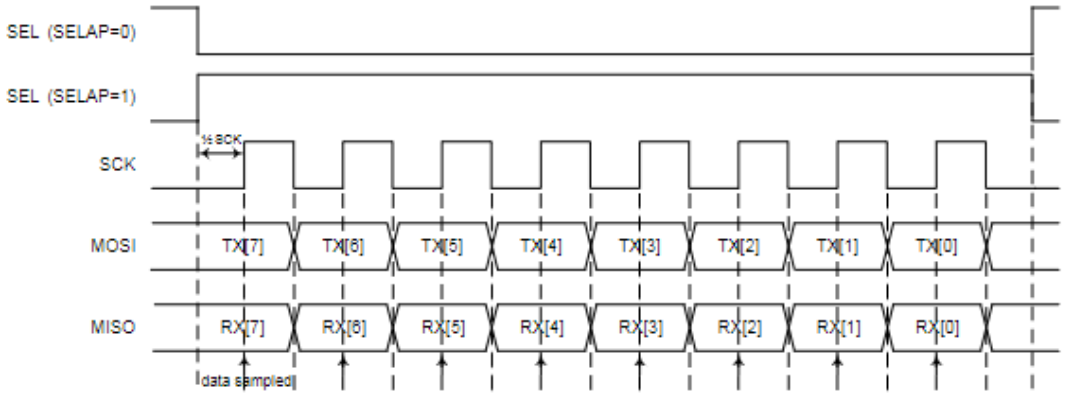
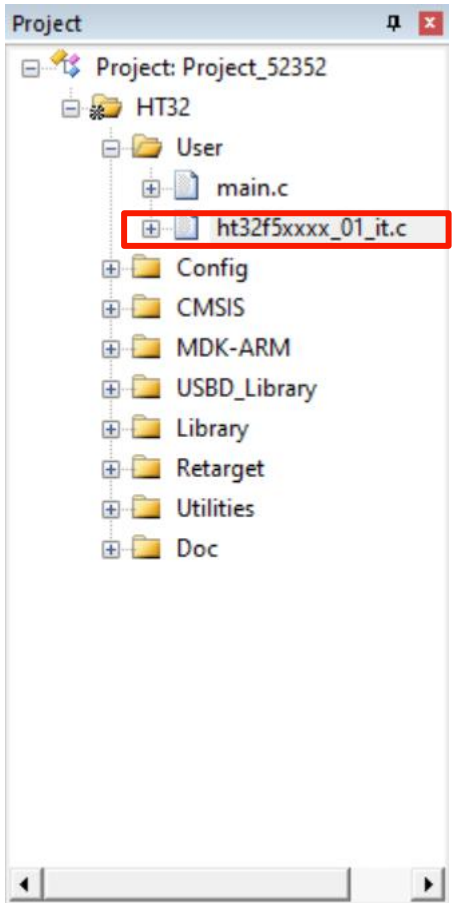


Figure 151. SPI Single Byte Transfer Timing Diagram – CPOL = 0, CPHA = 0

SPI Interrupt



Master Interrupt

```c
void HTCFG_SPI_MASTER_IRQHandler(void)
{
  if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_RXBNE))
  {
    /* Read received data
    SPI0_Buffer_Rx[SPI0_Rx_Index++] = SPI_ReceiveData(HTCFG_SPI_MASTER);
  }

  if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_TXBE))
  {
    if (SPI0_Tx_Index < BufferSize)
    {
      /* Send SPI0 data
      SPI_SendData(HTCFG_SPI_MASTER, SPI0_Buffer_Tx[SPI0_Tx_Index++]);
    }
    else
    {
      /* Disable SPI0 TXBE interrupt
      SPI_IntConfig(HTCFG_SPI_MASTER, SPI_INT_TXBE, DISABLE);
    }
  }
}
```

Slave Interrupt

```c
void HTCFG_SPI_SLAVE_IRQHandler(void)
{
  if (SPI_GetFlagStatus(HTCFG_SPI_SLAVE, SPI_FLAG_RXBNE))
  {
    /* Read received data
    SPI1_Buffer_Rx[SPI1_Rx_Index++] = SPI_ReceiveData(HTCFG_SPI_SLAVE);
  }

  if (SPI_GetFlagStatus(HTCFG_SPI_SLAVE, SPI_FLAG_TXBE))
  {
    if (SPI1_Tx_Index < BufferSize)
    {
      /* Send SPI1 data
      SPI_SendData(HTCFG_SPI_SLAVE, SPI1_Buffer_Tx[SPI1_Tx_Index++]);
    }
    else
    {
      /* Disable SPI1 TXBE interrupt
      SPI_IntConfig(HTCFG_SPI_SLAVE, SPI_INT_TXBE, DISABLE);
    }
  }
}
```

SPI Loopback

```c
void SPI_Loopback(void)
{
  /* Wait for transmission finished                                          */
  while (SPI0_Rx_Index < BufferSize);

  /* Check on validity of received data on SPI0 & SPI1                       */
  if (CmpBuffer(SPI0_Buffer_Tx, SPI1_Buffer_Rx, BufferSize) && CmpBuffer(SPI1_Buffer_Tx, SPI0_Buffer_Rx, BufferSize))
  {
    /* Turn on LED1 if the transmitted and received data are equal           */
    HT32F_DVB_LEDOn(HT_LED1);
  }
  else
  {
    /* Turn on LED2 if the transmitted and received data are different       */
    HT32F_DVB_LEDOn(HT_LED2);
  }
}
```

If SPI0 hasn't completed reception, do not proceed further

If the transmitted data isn't correct, LED2 will light up.
SPI0_TX ≠ SPI1_RX,
SPI1_TX ≠ SPI0_RX.

If the transmitted data is correct, LED1 will light up.
SPI0_TX = SPI1_RX,
SPI1_TX = SPI0_RX.

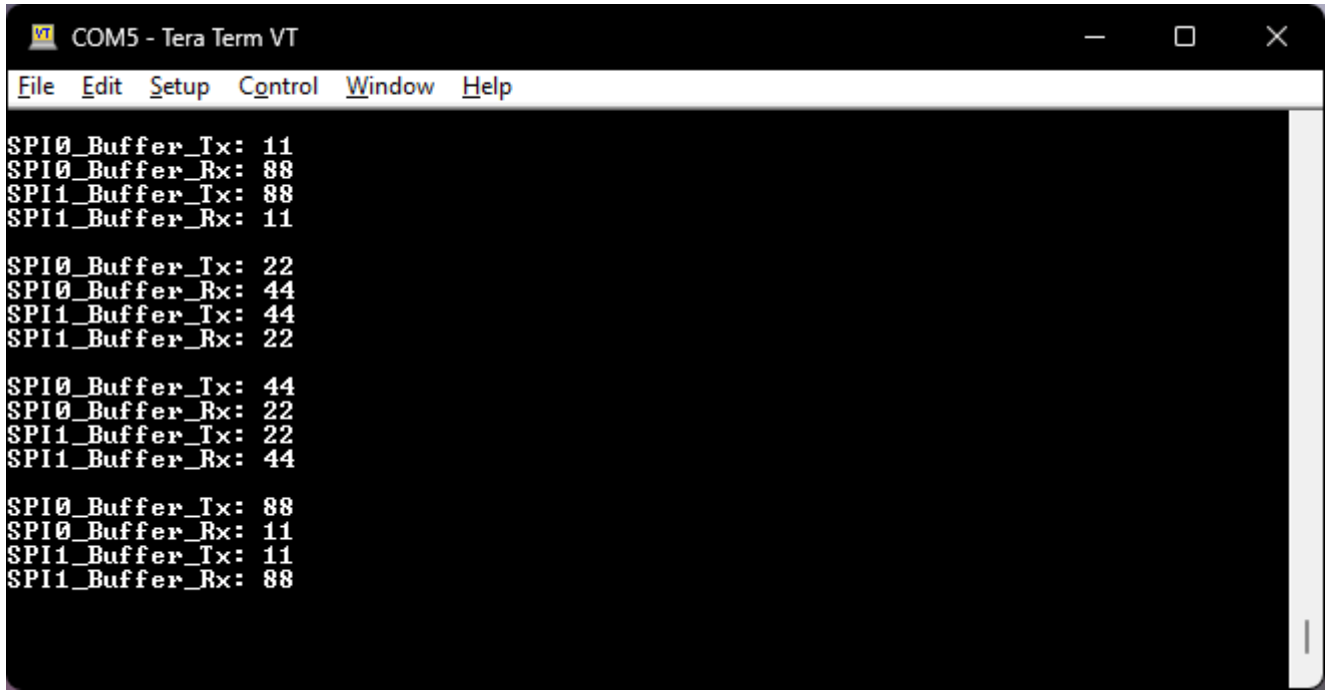
Homework W6-1.

https://github.com/CYCU-AIoT-System-Lab/Microcontroller-Experiment/blob/main/w6/SPI-Interrupt_Simulate-Experiment_Steps.md

Execute the example and show on Tera Term

- Objective: Wire and edit the code.
- Hint:

1. Use F12 to find out the pin
2. Wire. (Check p7)
3. Edit code.(Check p26)



Execute the example and show on Tera Term

```
RETARGET_Configuration();
```

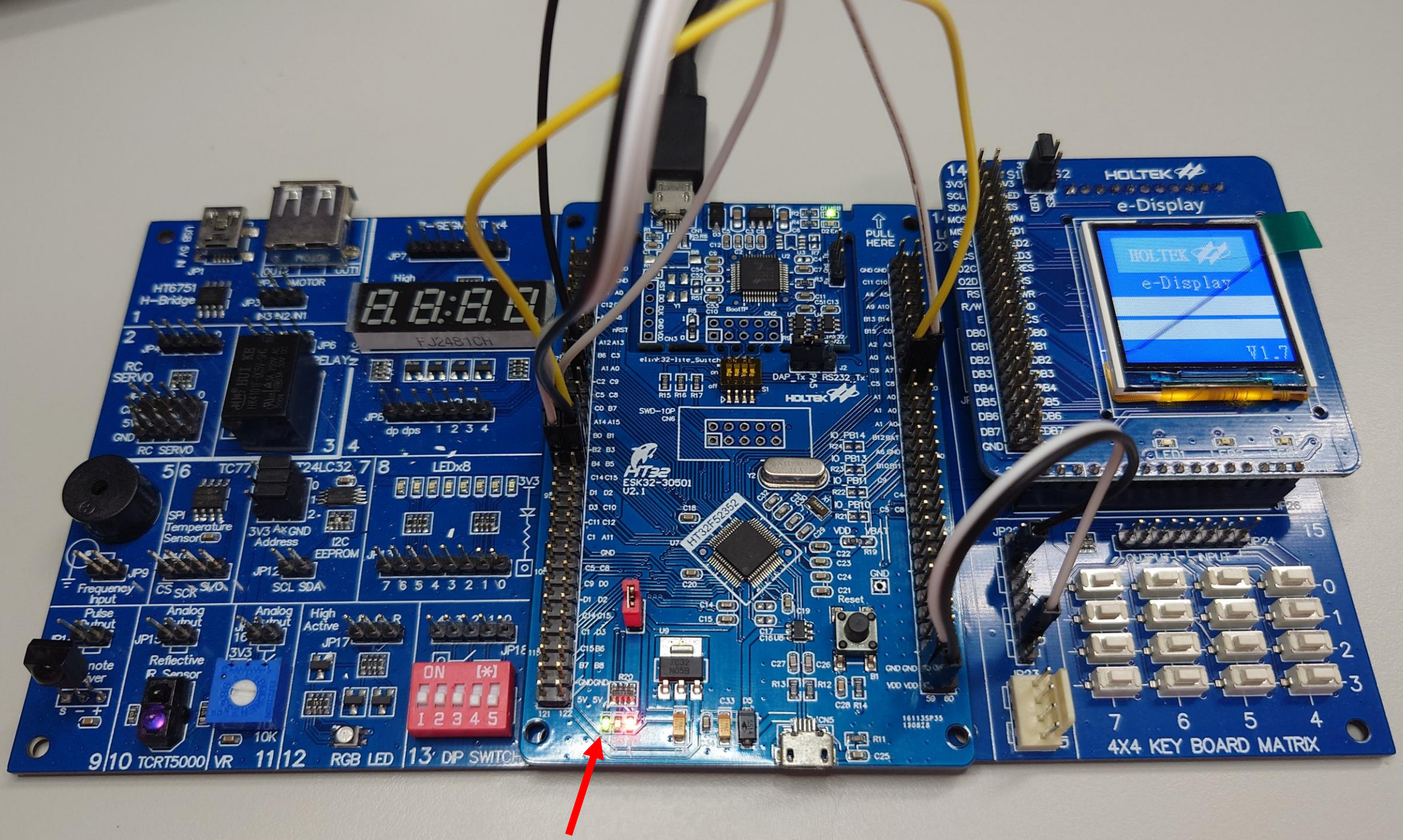
Add UART setting

```
void SPI_Loopback(void)
{
  vu32 i;
  /* Wait for transmission finished                                          */
  while (SPI0_Rx_Index < BufferSize);

  /* Check on validity of received data on SPI0 & SPI1                       */
  if (CmpBuffer(SPI0_Buffer_Tx, SPI1_Buffer_Rx, BufferSize) && CmpBuffer(SPI1_Buffer_Tx, SPI0_Buffer_Rx, BufferSize))
  {
    /* Turn on LED1 if the transmitted and received data are equal           */
    HT32F_DVB_LEDOn(HT_LED1);
  }
  else
  {
    /* Turn on LED2 if the transmitted and received data are different       */
    HT32F_DVB_LEDOn(HT_LED2);
  }
  for(i=0;i<BufferSize;i++)
  {
    printf("SPI0_Buffer_Tx: %x\r\n", SPI0_Buffer_Tx[i]);
    printf("SPI1_Buffer_Rx: %x\r\n", SPI1_Buffer_Tx[i]);
    printf("SPI1_Buffer_Tx: %x\r\n", SPI1_Buffer_Tx[i]);
    printf("SPI0_Buffer_Rx: %x\r\n", SPI0_Buffer_Rx[i]);
    printf("\n");
  }
}
```

Declare a variable for use in for loop

Display the data to check if the transmission was successful.



https://youtu.be/1A0dsZUfvKI

Homework W6-2

Read the temperature & show on Tera Term

- Objective: Use JP10 to read the temperature.
- Hint:

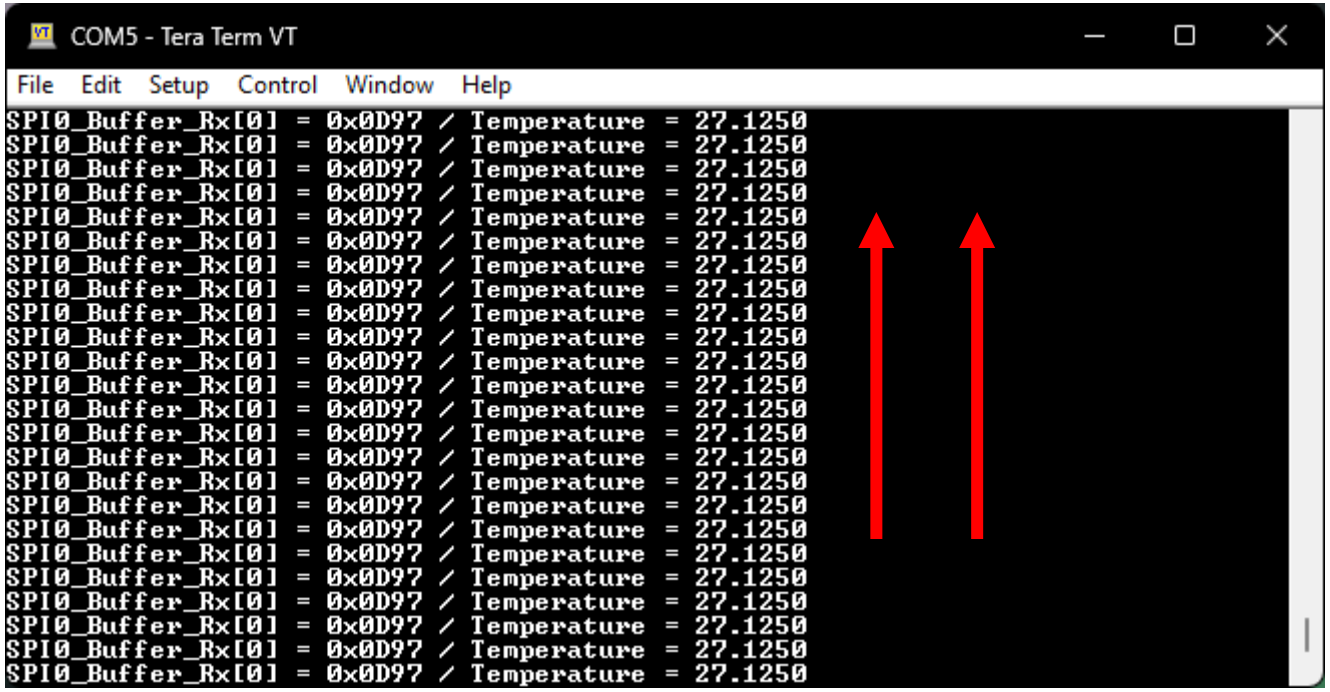
1. Wire:

   Master_SCK->SCK

   Master_SEL->CS

   Master_MISO->SI/O

2. Edit code.(P32、33)



3. Show the results on the TeraTerm. (number should change if you touch TC77)

TC77 temperature sensor module

Datasheet: http://ww1.microchip.com/downloads/en/devicedoc/20092a.pdf

Feature:

Temperature sensing range: -55°C~+125 °C

Number of output data bits: 16-bits

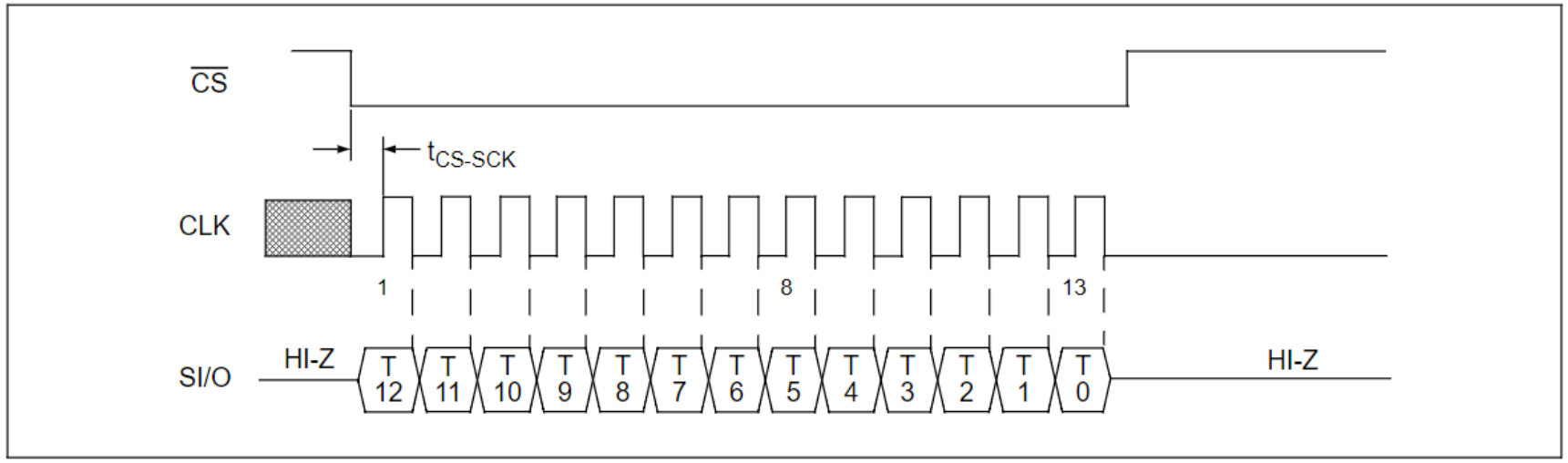


FIGURE 3-2: *Temperature Read Timing Diagram - (Reading only the first 13 Bits of the Temperature Register).*

TC77 output data format

The TC77 outputs 16 bits of data, with only 13 bits representing the valid data area.

- Bit15(sign): 1 is negative, 0 is positive
- Bit14~7(integer): Represent temperature values in binary.
- Bit6~3(decimal): Multiples of the decimal value 0.0625.
- Bit2~0(invalid)

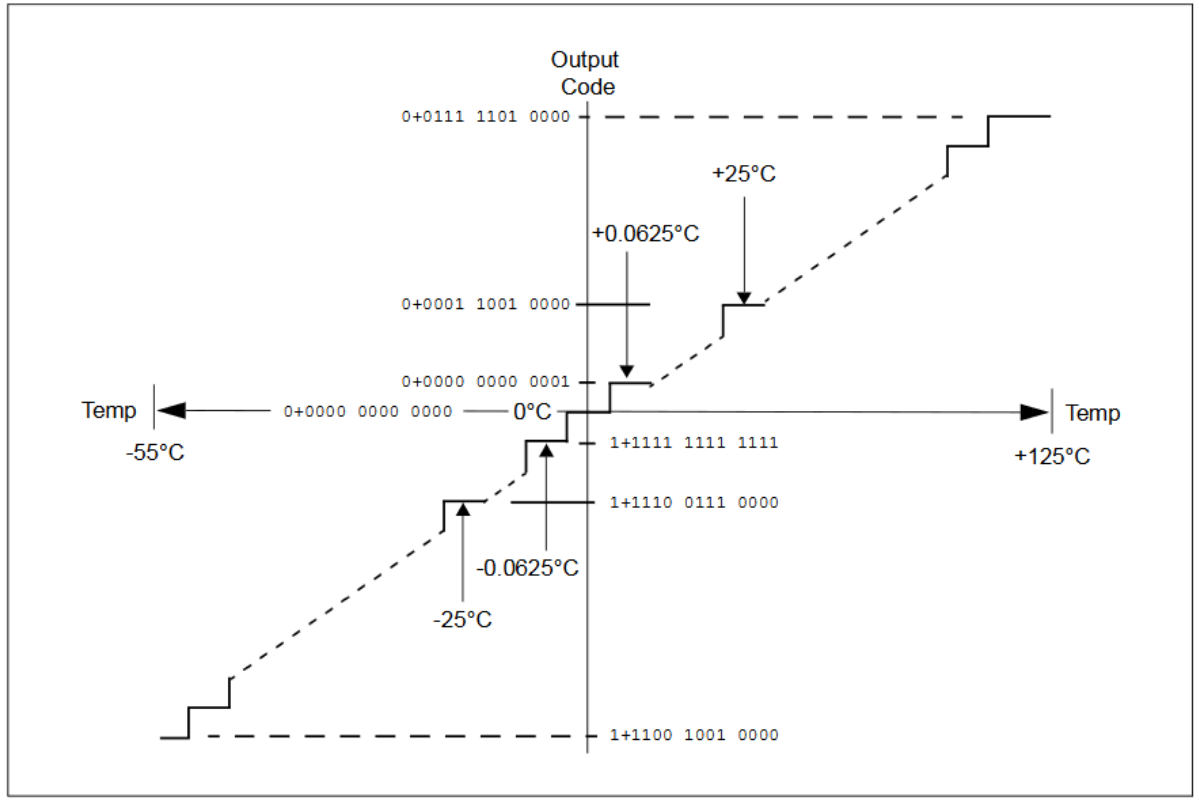


FIGURE 3-1: Temperature To Digital Transfer Function (Non-linear Scale).

Temperature	Binary	HEX
+125°C	0011 1110 1000 0111	3E87
+25°C	0000 1100 1000 0111	datasheet error 0CB7
+0.0625°C	0000 0000 0000 1111	000F
0°C	0000 0000 0000 0111	0007
-0.0625°C	1111 1111 1111 1111	FFFF
-25°C	1111 0011 1000 0111	F387
-55°C	1110 0100 1000 0111	E487

sign integer decimal invalid

main.c

1

```c
/* Private constants ---------------------
#define BufferSize          1
#define Temp_Int_Mask       0xFF
#define Temp_Dec_Mask       0x0F
#define TC77_LSB            0.0625f

/* Private function prototypes -----------
void SPI_Configuration(void);
float Convert_Temp(void);

/* Private variables ---------------------
u16 SPI0_Buffer_Rx[BufferSize] = {0};

```

2

```c
/* Global functions ----------------------------------------------------------
/*****************************************************************************
  * @brief  Main program.
  * @retval None
  *****************************************************************************
int main(void)
{
  RETARGET_Configuration();

  SPI_Configuration();

  while (1){
    printf("SPI0_Buffer_Rx[0] = 0x%04X / Temperature = %.4f\r\n", SPI0_Buffer_Rx[0], Convert_Temp());
  }
}
```

3

Remove SLAVE settings

```c
/*****************************************************************************
  * @brief  Configure the SPI ports.
  * @retval None
  *****************************************************************************
void SPI_Configuration(void)
{
  /* !!! NOTICE !!!
     Notice that the local variable (structure) did not have an initial value.
     Please confirm that there are no missing members in the parameter settings below in this function.
  */
  SPI_InitTypeDef SPI_InitStructure;

  CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
  /* Enable Px, Master & AFIO clock                                          */
  HTCFG_SPI_MASTER_SEL_GPIO_CLOCK(CKCUClock)          = 1;
  HTCFG_SPI_MASTER_CLOCK(CKCUClock)                   = 1;
  CKCUClock.Bit.AFIO                                  = 1;
  CKCU_PeripClockConfig(CKCUClock, ENABLE);

  /* MASTER_SEL idle state is HIGH
  GPIO_PullResistorConfig(HTCFG_SPI_MASTER_SEL_GPIO_ID, HTCFG_SPI_MASTER_SEL_AFIO_PIN, GPIO_PR_UP);

  /* Configure related IO to Master mode
  AFIO_GPxConfig(HTCFG_SPI_MASTER_SEL_AFIO_PORT, HTCFG_SPI_MASTER_SEL_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_MASTER_SCK_AFIO_PORT, HTCFG_SPI_MASTER_SCK_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_MASTER_MOSI_AFIO_PORT, HTCFG_SPI_MASTER_MOSI_AFIO_PIN, AFIO_FUN_SPI);
  AFIO_GPxConfig(HTCFG_SPI_MASTER_MISO_AFIO_PORT, HTCFG_SPI_MASTER_MISO_AFIO_PIN, AFIO_FUN_SPI);

  /* SPI configuration: Master mode                                          */
  SPI_InitStructure.SPI_Mode = SPI_MASTER;
  SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
  SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_16;
  SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
  SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
  SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
  SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
  SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
  SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
  SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
  SPI_InitStructure.SPI_ClockPrescaler = 2400;
  SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);

  /* Set SEL as output mode for slave select                                 */
  SPI_SELOutputCmd(HTCFG_SPI_MASTER, ENABLE);

  /* Enable SPI0 TXBE & RXBNE interrupt                                      */
  SPI_IntConfig(HTCFG_SPI_MASTER, SPI_INT_TXBE | SPI_INT_RXBNE, ENABLE);

  /* Configure and enable master interrupt                                   */
  NVIC_SetPriority(HTCFG_SPI_MASTER_IRQn, 0);
  NVIC_EnableIRQ(HTCFG_SPI_MASTER_IRQn);

  /* Enable master later                                                     */
  SPI_Cmd(HTCFG_SPI_MASTER, ENABLE);
}
```

SPI_DATALENGTH_16

2400

4

```c
/*****************************************************************************
  * @brief  Convert the temperature value to real temperature.
  * @retval None
  *****************************************************************************
float Convert_Temp(void)
{
  u16 temp_buffer = SPI0_Buffer_Rx[0];
  int temp_pol, temp_int, temp_dec;
  temp_pol = (temp_buffer >> 15) ? -1 : 1;
  temp_int = (temp_buffer >> 07) & Temp_Int_Mask;
  temp_dec = (temp_buffer >> 03) & Temp_Dec_Mask;
  if (temp_pol == -1){
    temp_int = Temp_Int_Mask - temp_int;
    temp_dec = Temp_Dec_Mask - temp_dec + 1;
  }
  return temp_pol * (temp_int + temp_dec * TC77_LSB);
}
```

ht32f5xxxx_01_it.c

1

```c
/* Private variables ---------
extern vu16 SPI0_Buffer_Rx[];
```

Use array defined in main.c

2

```c
/*****************************************************************************//**
 * @brief   This function handles SPI Master interrupt.
 * @retval  None
 *****************************************************************************/
void HTCFG_SPI_MASTER_IRQHandler(void)
{
  if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_RXBNE))
  {
    /* Read received data                                                    */
    SPI0_Buffer_Rx[0] = SPI_ReceiveData(HTCFG_SPI_MASTER);
  }

  if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_TXBE))
  {
    /* Send data                                                             */
    SPI_SendData(HTCFG_SPI_MASTER, 0x00);
  }
}
```

If the receive buffer of SP0 has data, then receive SPI0 data into the SPI1_Rx array.

If the transmit buffer of SPI1 has data, then send the data out.

3

```c
void HTCFG_SPI_SLAVE_IRQHandler(void)
{
}
```

Remove content in HTCFG_SPI_SLAVE_IRQHandler

Temperature conversion formula

The data read from TC77: SPI0_Buffer_Rx[0]

For example: 25.0625 °C

0x0C8F(TC77_hexadecimal)

0b0000 1100 1000 1111(TC77_binary)

0b0000 0000 0000 0000(SPI0_Buffer_Rx[0]>>15) →0 →+

0b0000 0000 0001 1001(SPI0_Buffer_Rx[0]>>7) →25

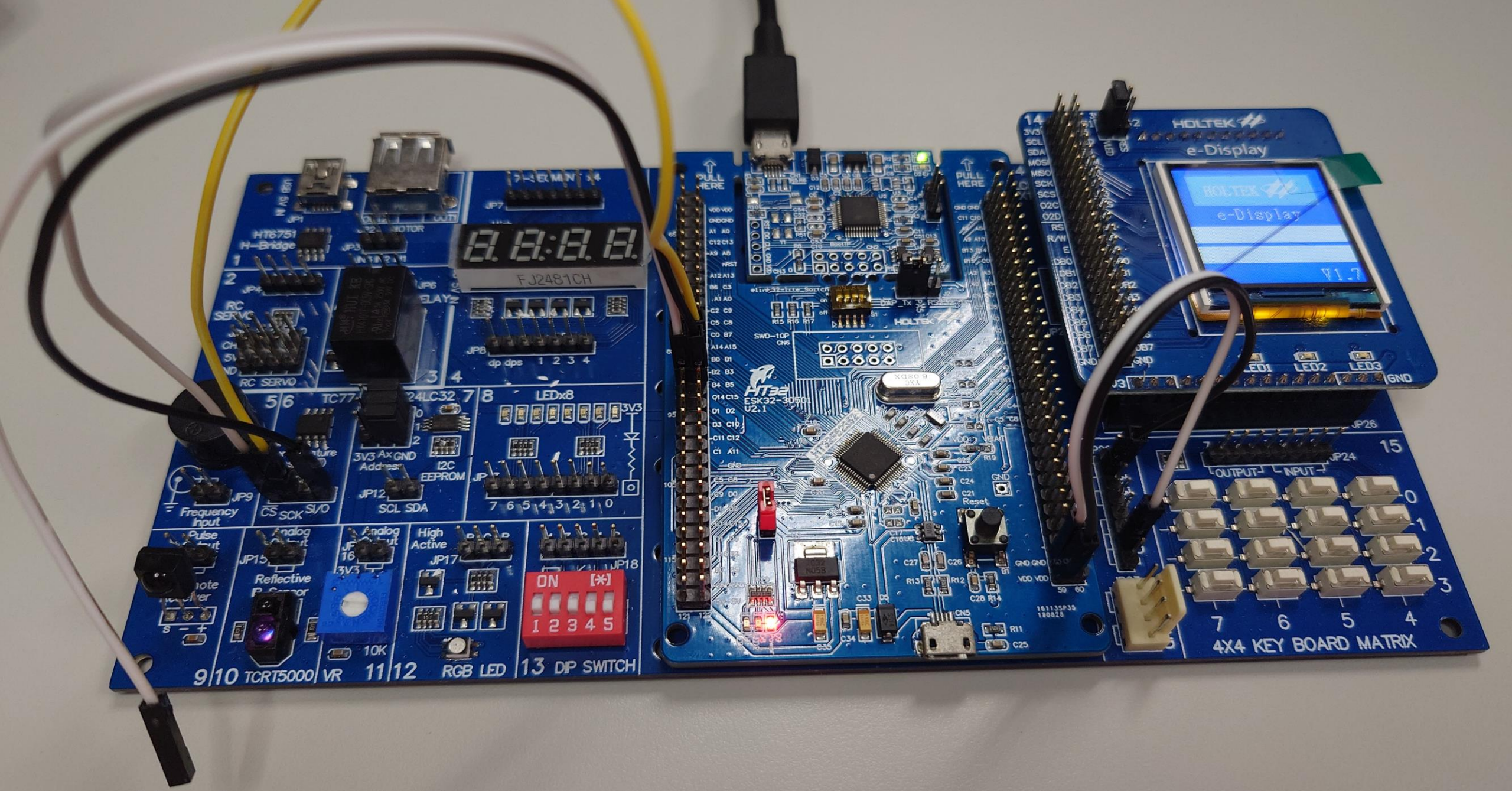
0b0000 0000 0001 1001(SPI0_Buffer_Rx[0]>>7)&0xFF →25

0b0000 0001 1001 0001(SPI0_Buffer_Rx[0]>>3) →402

0b0000 0001 1001 0001(SPI0_Buffer_Rx[0]>>3)&0xF →1

0b0000 0001 1001 0001(SPI0_Buffer_Rx[0]>>3)&0xF*0.0625 →0.0625

+ → 25.0625



https://youtu.be/alLRYzyeQVI

Class Dismissed