

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

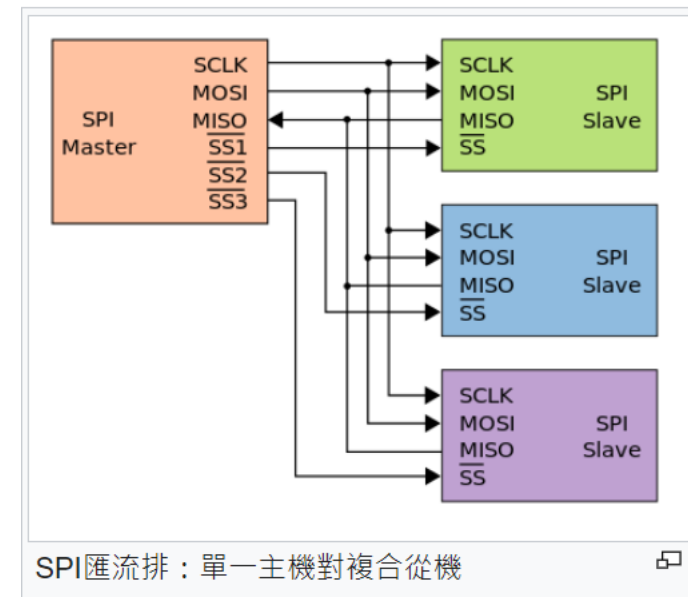
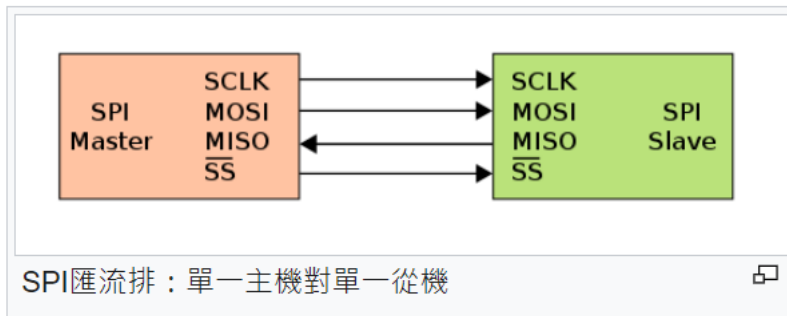
1. SPI introduction
2. SPI Project.
3. Homework 6-1.
4. Homework 6-2.

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, containing the text 'SPI Introduction'.

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.



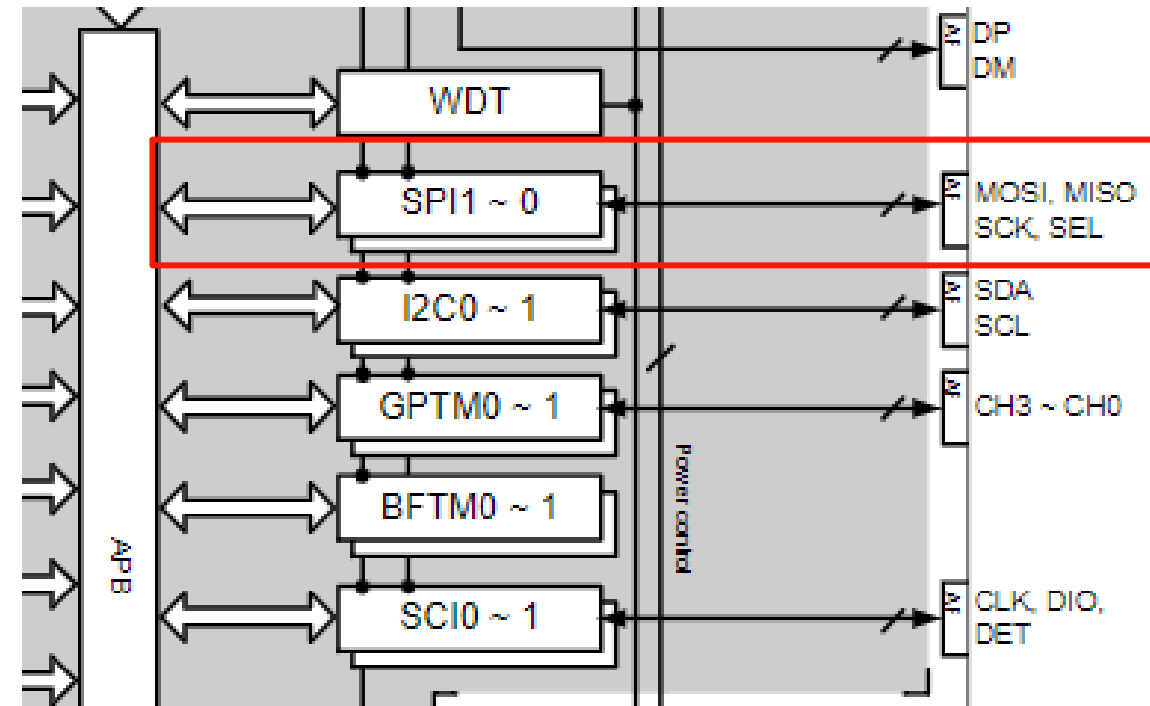
Operation

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)



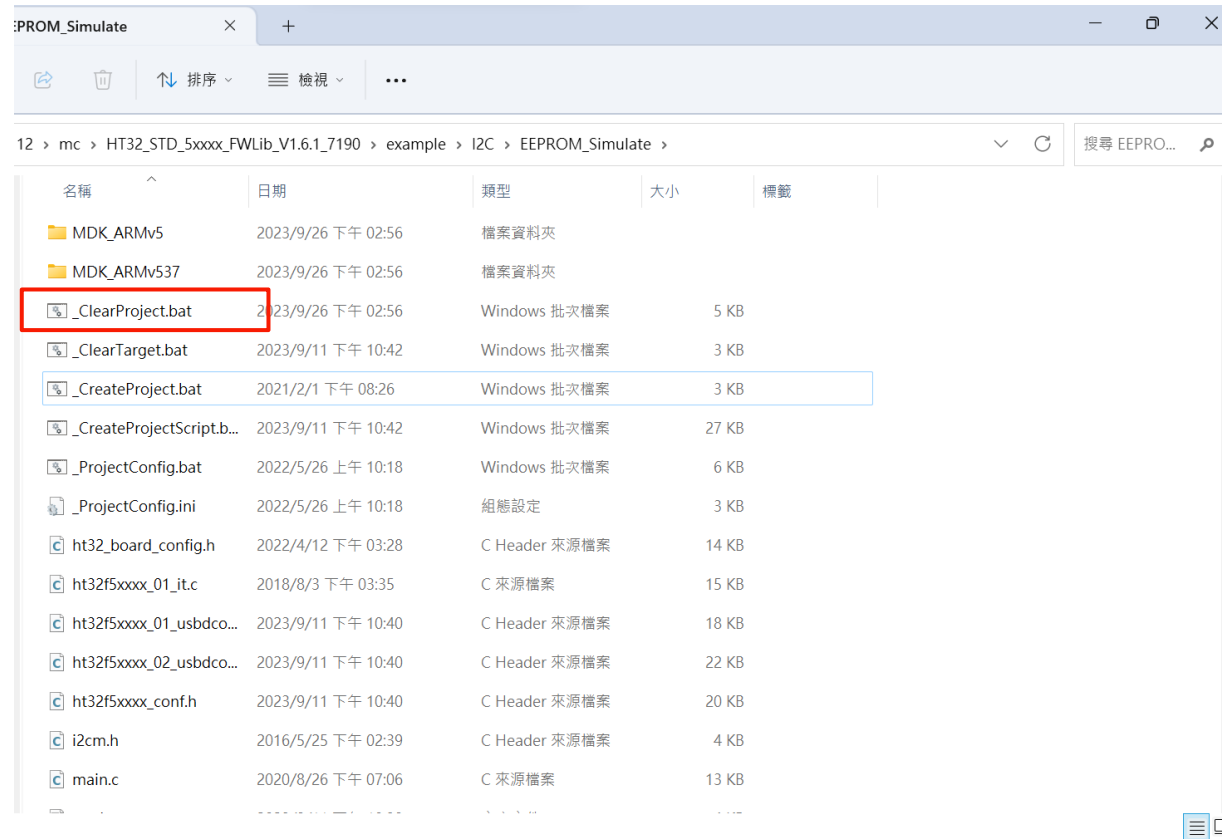
The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large, solid red speech bubble is centered on the page, pointing downwards.

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

Figure 2 illustrates the process of launching a project in the μVision IDE. The left panel shows a file explorer view of the project directory, highlighting the **MDK_ARMv537** folder and the **Project_52352.uvprojx** file. The right panel shows the μVision IDE interface with the **main.c** file open, displaying the project's source code and the build output window.

The build output window shows the following error message:

```
linking...
Program Size: Code=1868 RO-data=68 RW-data=8 ZI-data=528
FromELF: creating hex file...
After Build - User command #1: fromelf --bin -o ".\HT32\52352\Obj\HT32.axf.bin" ".\HT32\52352\Obj\HT32.axf"
".\HT32\52352\Obj\HT32.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:15
```

main

```
70 int main(void)
71 {
72     /* Initialize LED1 & LED2 on HT32 board
73     HT32F_DVB_LEDInit(HT_LED1);
74     HT32F_DVB_LEDInit(HT_LED2);
75
76     SPI_Configuration();
77
78     SPI_Loopback();
79
80     while (1);
81 }
```

LED Initialization Setting

SPI Initialization Setting

SPI Return Function Testing

SPI Initialization Setting

```
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 function.
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_UP);
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);
111
112     /* Configure related IO to Slave mode
113     AFIO_GPxConfig(HTCFG_SPI_SLAVE_SEL_AFIO_PORT, HTCFG_SPI_SLAVE_SEL_AFIO_PIN, AFIO_FUN_SPI);
114     AFIO_GPxConfig(HTCFG_SPI_SLAVE_SCK_AFIO_PORT, HTCFG_SPI_SLAVE_SCK_AFIO_PIN, AFIO_FUN_SPI);
115     AFIO_GPxConfig(HTCFG_SPI_SLAVE_MOSI_AFIO_PORT, HTCFG_SPI_SLAVE_MOSI_AFIO_PIN, AFIO_FUN_SPI);
116     AFIO_GPxConfig(HTCFG_SPI_SLAVE_MISO_AFIO_PORT, HTCFG_SPI_SLAVE_MISO_AFIO_PIN, AFIO_FUN_SPI);
117
118     /* SPI configuration: Master mode
119     SPI_InitStructure.SPI_Mode = SPI_MASTER;
120     SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
121     SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_8;
122     SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
123     SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
124     SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
125     SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
126     SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
127     SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
128     SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
129     SPI_InitStructure.SPI_ClockPrescaler = 4;
130     SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);
131
132     /* SPI configuration: Slave mode
133     SPI_InitStructure.SPI_Mode = SPI_SLAVE;
134     SPI_Init(HTCFG_SPI_SLAVE, &SPI_InitStructure);
135 }
```

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_A, HTCFG_SPI_MASTER_MOSI_AFIO_PIN, AFIO_FUN_SPI);
110     AFIO_GPxConfig(HTCFG_SPI_MASTER_MISO_A, HTCFG_SPI_MASTER_MISO_AFIO_PIN, AFIO_FUN_SPI);

```

```

137 #define HTCFG_SPI_MASTER_CLOCK(CK) (CK.Bit.SPI0)
138 #define HTCFG_SPI_MASTER_SEL_GPIO_ID (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

```
118 /* SPI configuration: Master mode
119 SPI_InitStructure.SPI_Mode = SPI_MASTER;
120 SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
121 SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_8;
122 SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
123 SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
124 SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
125 SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
126 SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
127 SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
128 SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
129 SPI_InitStructure.SPI_ClockPrescaler = 4;
130 SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);
```

F12

```
138 void SPI_Init(HT_SPI_TypeDef* SPIx, SPI_InitTypeDef* SPI_InitStruct)
139 {
140     u32 tmp;
141
142     /* Check the parameters */
143     Assert_Param(IS_SPI(SPIx));
144     Assert_Param(IS_SPI_MODE(SPI_InitStruct->SPI_Mode));
145     Assert_Param(IS_SPI_FIFO_SET(SPI_InitStruct->SPI_FIFO));
146     Assert_Param(IS_SPI_DATALENGTH(SPI_InitStruct->SPI_DataLength));
147     Assert_Param(IS_SPI_SEL_MODE(SPI_InitStruct->SPI_SELMode));
148     Assert_Param(IS_SPI_SEL_POLARITY(SPI_InitStruct->SPI_SELPolarity));
149     Assert_Param(IS_SPI_CPOL(SPI_InitStruct->SPI_CPOL));
150     Assert_Param(IS_SPI_CPHA(SPI_InitStruct->SPI_CPHA));
151     Assert_Param(IS_SPI_FIRST_BIT(SPI_InitStruct->SPI_FirstBit));
152     Assert_Param(IS_SPI_FIFO_LEVEL(SPI_InitStruct->SPI_RxFIFOTriggerLevel));
153     Assert_Param(IS_SPI_FIFO_LEVEL(SPI_InitStruct->SPI_TxFIFOTriggerLevel));
154     Assert_Param(IS_SPI_CLOCK_PRESCALER(SPI_InitStruct->SPI_ClockPrescaler));
155
156     /*----- SPIx Control Register 2 Configuration -----*/
157     tmp = SPI_InitStruct->SPI_CPOL;
158     if (tmp == SPI_CPOL_LOW)
159     {
160         tmp |= (0x100 << SPI_InitStruct->SPI_CPHA);
161     }
162     else
163     {
164         tmp |= (0x200 >> SPI_InitStruct->SPI_CPHA);
165     }
166
167     SPIx->CR1 = SPI_InitStruct->SPI_Mode | SPI_InitStruct->SPI_DataLength |
168                SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
169                SPI_InitStruct->SPI_FirstBit | tmp;
170
171     /*----- SPIx FIFO Control Register Configuration -----*/
172     SPIx->FCR = SPI_InitStruct->SPI_FIFO | SPI_InitStruct->SPI_TxFIFOTriggerLevel |
173                (SPI_InitStruct->SPI_RxFIFOTriggerLevel << 4);
174
175     /*----- SPIx Clock Prescaler Register Configuration -----*/
176     #if (LIBCFG_SPI_CLK_PRE_V01)
177     SPIx->CPR = (SPI_InitStruct->SPI_ClockPrescaler - 1);
178     #else
179     SPIx->CPR = (SPI_InitStruct->SPI_ClockPrescaler / 2) - 1;
180     #endif
181 }
```

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

SPI FIFO Control Register

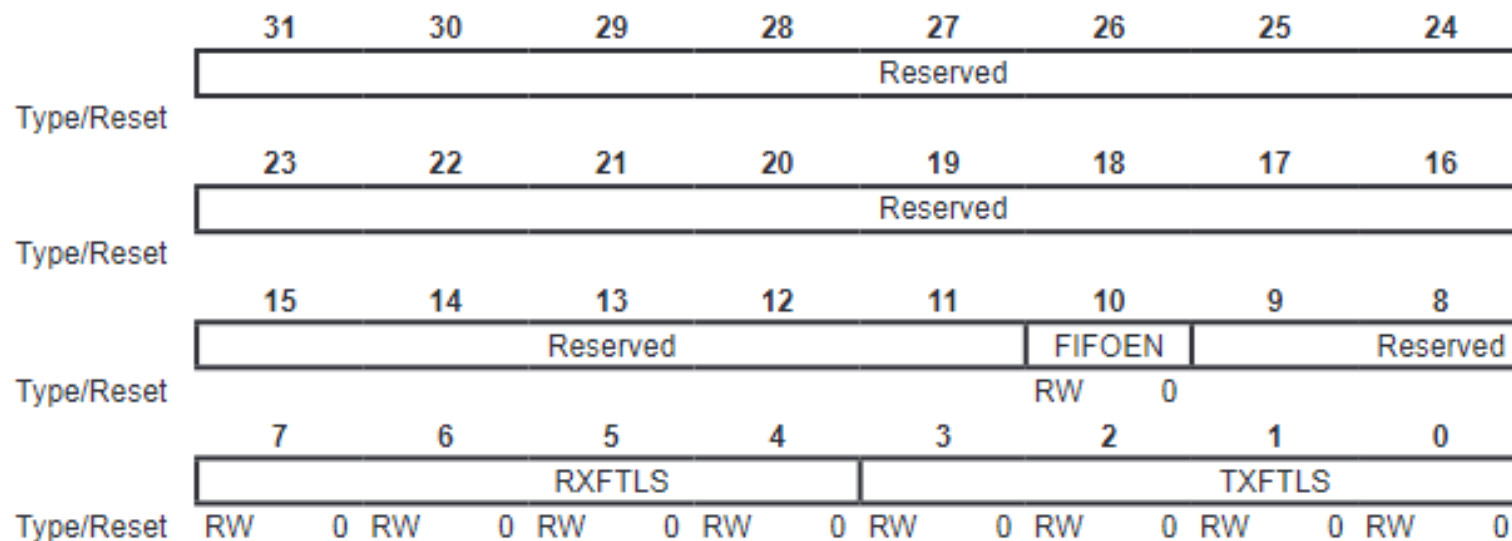
➤ UserManual P495

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



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		
Type/Reset	Reserved									
	23	22	21	20	19	18	17	16		
Type/Reset	Reserved									
	15	14	13	12	11	10	9	8		
Type/Reset	CP									
	RW	0	RW	0	RW	0	RW	0	RW	0
	7	6	5	4	3	2	1	0		
Type/Reset	CP									
	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:

```
156  /*----- SPIx Control Register 2 Configuration -----*/
157  tmp = SPI_InitStruct->SPI_CPOL;
158  if (tmp == SPI_CPOL_LOW)
159  {
160      tmp |= (0x100 << SPI_InitStruct->SPI_CPHA);
161  }
162  else
163  {
164      tmp |= (0x200 >> SPI_InitStruct->SPI_CPHA);
165  }
166
167  SPIx->CR1 = SPI_InitStruct->SPI_Mode | SPI_InitStruct->SPI_DataLength |
168             SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
169             SPI_InitStruct->SPI_FirstBit | tmp;
```

Print on the Tera Term to confirm calculation result

```
70 int main(void)
71 {
72     RETARGET_Configuration();
73     /* Initialize LED1 & LED2 on HT32 board
74     HT32F_DVB_LEDInit(HT_LED1);
75     HT32F_DVB_LEDInit(HT_LED2);
76
77     SPI_Configuration();
78
79     SPI_Loopback();
80
81     while (1);
82 }
```

```
156 /*----- SPIx Control Register 2 Cor
157 tmp = SPI_InitStruct->SPI_CPOL;
158 if (tmp == SPI_CPOL_LOW)
159 {
160     tmp |= (0x100 << SPI_InitStruct->SPI_CPHA);
161 }
162 else
163 {
164     tmp |= (0x200 >> SPI_InitStruct->SPI_CPHA);
165 }
166
167 SPIx->CR1 = SPI_InitStruct->SPI_Mode | SPI_InitStruct->SPI_DataLength |
168             SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
169             SPI_InitStruct->SPI_FirstBit | tmp;
170 printf("CR1: %10x\r\n", SPI_InitStruct->SPI_Mode | SPI_InitStruct->SPI_DataLength |
171        SPI_InitStruct->SPI_SELMode | SPI_InitStruct->SPI_SELPolarity |
172        SPI_InitStruct->SPI_FirstBit | tmp);
```

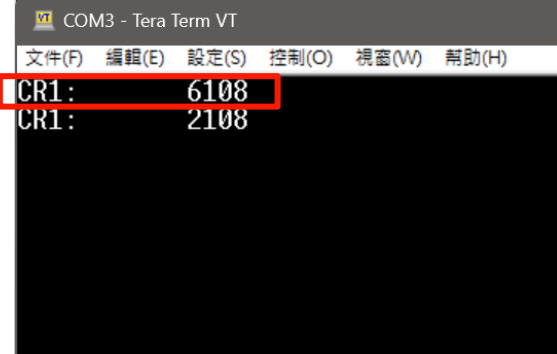
SPI Control Register 1 – SPICR1

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

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

```
119  /* SPI configuration: Master mode
120  SPI_InitStructure.SPI_Mode = SPI_MASTER;
121  SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
122  SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_8;
123  SPI_InitStructure.SPI_SELMode = SPI_SEL_HARDWARE;
124  SPI_InitStructure.SPI_SELPolarity = SPI_SELPOLARITY_LOW;
125  SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
126  SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
127  SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
128  SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
129  SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
130  SPI_InitStructure.SPI_ClockPrescaler = 4;
131  SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);
```



CPOL = 0
CPHA = 0

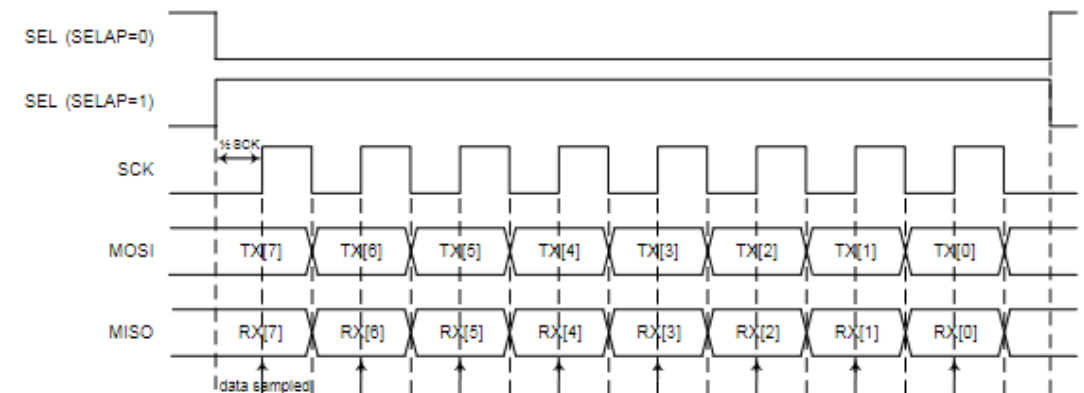
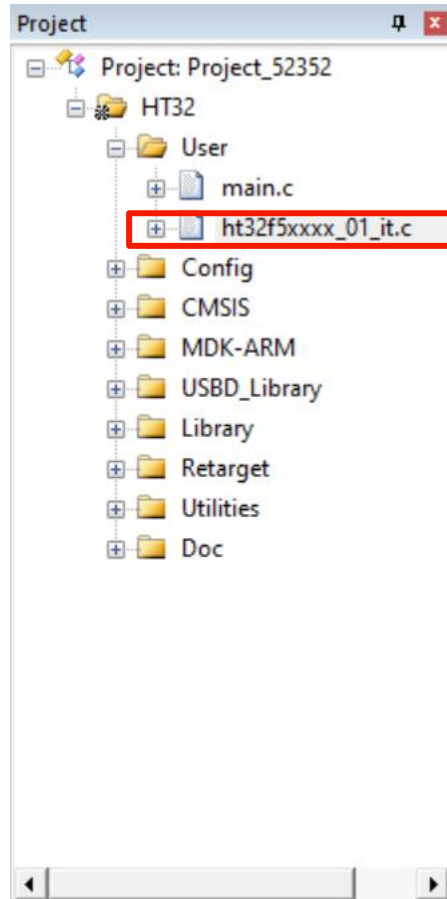


Figure 151. SPI Single Byte Transfer Timing Diagram – CPOL = 0, CPHA = 0

SPI Interrupt



Master
Interrupt

```
125 void HTCFCG_SPI_MASTER_IRQHandler(void)
126 {
127     if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_RXBNE))
128     {
129         /* Read received data
130         SPI0_Buffer_Rx[SPI0_Rx_Index++] = SPI_ReceiveData(HTCFG_SPI_MASTER);
131     }
132
133     if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_TXBE))
134     {
135         if (SPI0_Tx_Index < BufferSize)
136         {
137             /* Send SPI0 data
138             SPI_SendData(HTCFG_SPI_MASTER, SPI0_Buffer_Tx[SPI0_Tx_Index++]);
139         }
140         else
141         {
142             /* Disable SPI0 TXBE interrupt
143             SPI_IntConfig(HTCFG_SPI_MASTER, SPI_INT_TXBE, DISABLE);
144         }
145     }
146 }
```

Slave
Interrupt

```
152 void HTCFCG_SPI_SLAVE_IRQHandler(void)
153 {
154     if (SPI_GetFlagStatus(HTCFG_SPI_SLAVE, SPI_FLAG_RXBNE))
155     {
156         /* Read received data
157         SPI1_Buffer_Rx[SPI1_Rx_Index++] = SPI_ReceiveData(HTCFG_SPI_SLAVE);
158     }
159
160     if (SPI_GetFlagStatus(HTCFG_SPI_SLAVE, SPI_FLAG_TXBE))
161     {
162         if (SPI1_Tx_Index < BufferSize)
163         {
164             /* Send SPI1 data
165             SPI_SendData(HTCFG_SPI_SLAVE, SPI1_Buffer_Tx[SPI1_Tx_Index++]);
166         }
167         else
168         {
169             /* Disable SPI1 TXBE interrupt
170             SPI_IntConfig(HTCFG_SPI_SLAVE, SPI_INT_TXBE, DISABLE);
171         }
172     }
173 }
```

SPI Loopback

```
163 void SPI_Loopback(void)
164 {
165     /* Wait for transmission finished */
166     while (SPI0_Rx_Index < BufferSize);
167
168     /* Check on validity of received data on SPI0 & SPI1 */
169     if (CmpBuffer(SPI0_Buffer_Tx, SPI1_Buffer_Rx, BufferSize) && CmpBuffer(SPI1_Buffer_Tx, SPI0_Buffer_Rx, BufferSize))
170     {
171         /* Turn on LED1 if the transmitted and received data are equal */
172         HT32F_DVB_LEDOn(HT_LED1);
173     }
174     else
175     {
176         /* Turn on LED2 if the transmitted and received data are different */
177         HT32F_DVB_LEDOn(HT_LED2);
178     }
179 }
```

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

If the transmitted data isn't correct, LED2 will light up.
SPI0_TX \neq SPI1_RX,
SPI1_TX \neq 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](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)



The screenshot shows a Tera Term VT window titled 'COM5 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main display area shows the following text:

```
SPI0_Buffer_Tx: 11
SPI0_Buffer_Rx: 88
SPI1_Buffer_Tx: 88
SPI1_Buffer_Rx: 11

SPI0_Buffer_Tx: 22
SPI0_Buffer_Rx: 44
SPI1_Buffer_Tx: 44
SPI1_Buffer_Rx: 22

SPI0_Buffer_Tx: 44
SPI0_Buffer_Rx: 22
SPI1_Buffer_Tx: 22
SPI1_Buffer_Rx: 44

SPI0_Buffer_Tx: 88
SPI0_Buffer_Rx: 11
SPI1_Buffer_Tx: 11
SPI1_Buffer_Rx: 88
```

Execute the example and show on Tera Term

```
RETARGET_Configuration();
```

→ Add UART setting

```
void SPI_Loopback(void)
```

```
{
```

```
    vu32 i;
```

→ Declare a variable for use in for loop

```
    /* 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]);
```

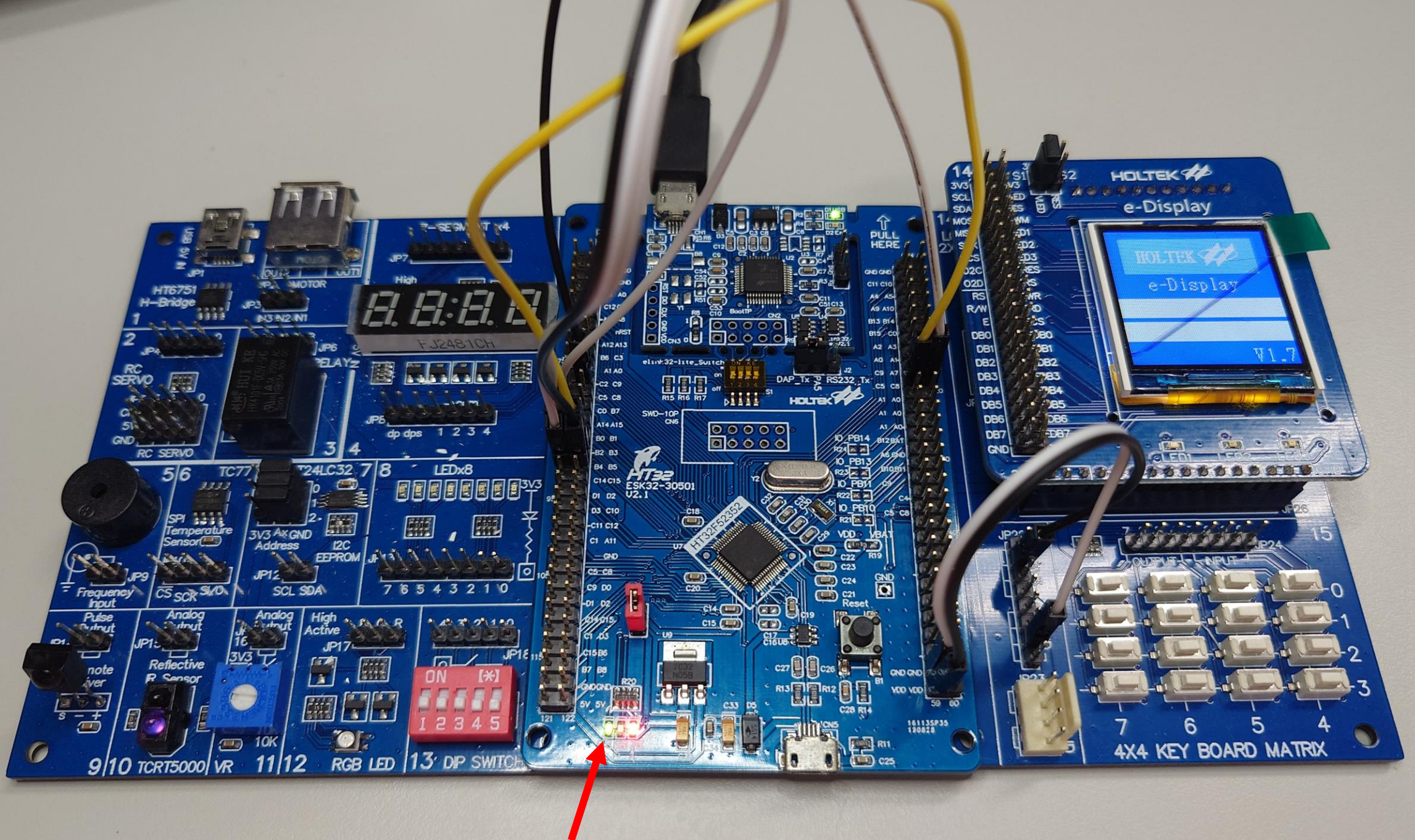
```
        printf("SPI0_Buffer_Rx: %x\r\n", SPI0_Buffer_Rx[i]);
```

```
        printf("\n");
```

```
    }
```

```
}
```

→ Display the data to check if the transmission was successful.



The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. In the center, there is a red speech bubble with a white outline. The text "Homework W6-2" is written in white inside the bubble.

Homework W6-2

Read the temperature & show on Tera Term

- Objective: Use JP10 to read the temperature.

- Hint:

1. Wire:

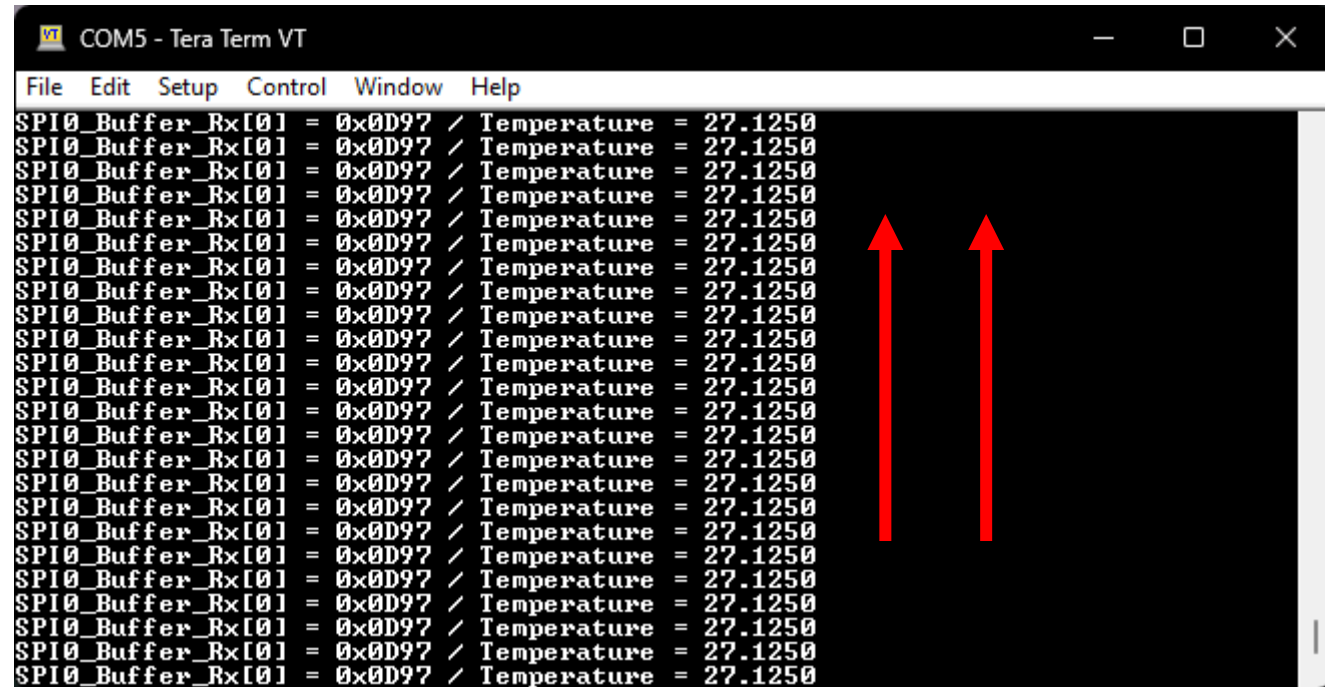
Master_SCK->SCK

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)



The screenshot shows a Tera Term window titled 'COM5 - Tera Term VT'. The window displays a continuous stream of text: 'SPI0_Buffer_Rx[0] = 0x0D97 / Temperature = 27.1250'. This line is repeated 20 times. Two red arrows point upwards from the bottom of the window towards the 'Temperature' values in the text stream.

TC77 temperature sensor module

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

Feature:

Temperature sensing range: $-55^{\circ}\text{C} \sim +125^{\circ}\text{C}$

Number of output data bits: 16-bits

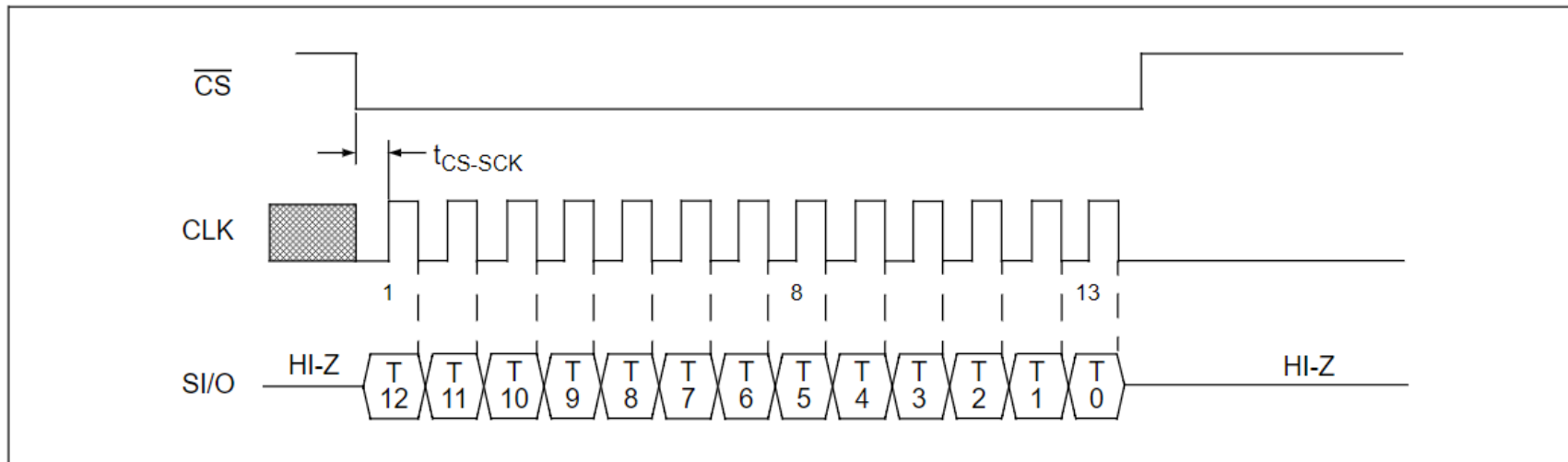


FIGURE 3-2: Temperature Read Timing Diagram - (Reading only the first 13 Bits of the Temperature Register).

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)

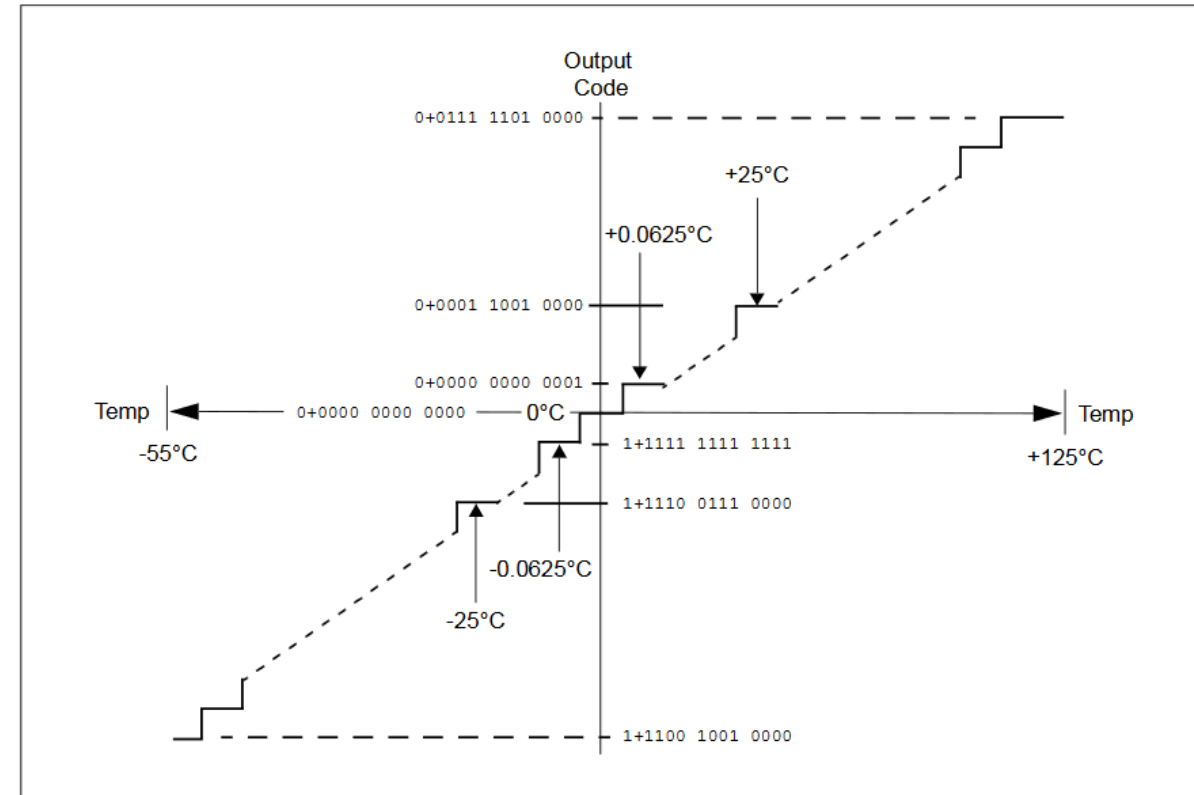


FIGURE 3-1: Temperature To Digital Transfer Function (Non-linear Scale).

Temperature	Binary	HEX
+125°C	0011 1110 1000 0111	3E87
+25°C	0000 1100 1000 0111	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

datasheet
error

main.c

```
74 /* *****//**
75  * @brief Configure the SPI ports.
76  * @retval None
77  * *****//
78 void SPI_Configuration(void)
79 {
80     /* !!! NOTICE !!!
81      * Notice that the local variable (structure) did not have an initial value.
82      * Please confirm that there are no missing members in the parameter settings below in this function.
83      */
84     SPI_InitTypeDef SPI_InitStructure;
85
86     CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
87     /* Enable Px, Master & AFIO clock
88     HTCFG_SPI_MASTER_SEL_GPIO_CLOCK(CKCUClock) = 1;
89     HTCFG_SPI_MASTER_CLOCK(CKCUClock) = 1;
90     CKCUClock.Bit.AFIO = 1;
91     CKCU_PeripClockConfig(CKCUClock, ENABLE);
92
93     /* MASTER_SEL idle state is HIGH
94     GPIO_PullResistorConfig(HTCFG_SPI_MASTER_SEL_GPIO_ID, HTCFG_SPI_MASTER_SEL_AFIO_PIN, GPIO_PR_UP);
95
96     /* Configure related IO to Master mode
97     AFIO_GPxConfig(HTCFG_SPI_MASTER_SEL_AFIO_PORT, HTCFG_SPI_MASTER_SEL_AFIO_PIN, AFIO_FUN_SPI);
98     AFIO_GPxConfig(HTCFG_SPI_MASTER_SCK_AFIO_PORT, HTCFG_SPI_MASTER_SCK_AFIO_PIN, AFIO_FUN_SPI);
99     AFIO_GPxConfig(HTCFG_SPI_MASTER_MOSI_AFIO_PORT, HTCFG_SPI_MASTER_MOSI_AFIO_PIN, AFIO_FUN_SPI);
00     AFIO_GPxConfig(HTCFG_SPI_MASTER_MISO_AFIO_PORT, HTCFG_SPI_MASTER_MISO_AFIO_PIN, AFIO_FUN_SPI);
01
02     /* SPI configuration: Master mode
03     SPI_InitStructure.SPI_Mode = SPI_MASTER;
04     SPI_InitStructure.SPI_FIFO = SPI_FIFO_DISABLE;
05     SPI_InitStructure.SPI_DataLength = SPI_DATALENGTH_16;
06     SPI_InitStructure.SPI_SelMode = SPI_SEL_HARDWARE;
07     SPI_InitStructure.SPI_SelPolarity = SPI_SELPOLARITY_LOW;
08     SPI_InitStructure.SPI_CPOL = SPI_CPOL_LOW;
09     SPI_InitStructure.SPI_CPHA = SPI_CPHA_FIRST;
10     SPI_InitStructure.SPI_FirstBit = SPI_FIRSTBIT_MSB;
11     SPI_InitStructure.SPI_RxFIFOTriggerLevel = 0;
12     SPI_InitStructure.SPI_TxFIFOTriggerLevel = 0;
13     SPI_InitStructure.SPI_ClockPrescaler = 2400;
14     SPI_Init(HTCFG_SPI_MASTER, &SPI_InitStructure);
15
16     /* Set SEL as output mode for slave select
17     SPI_SELOutputCmd(HTCFG_SPI_MASTER, ENABLE);
18
19     /* Enable SPI0 TXBE & RXBNE interrupt
20     SPI_IntConfig(HTCFG_SPI_MASTER, SPI_INT_TXBE | SPI_INT_RXBNE, ENABLE);
21
22     /* Configure and enable master interrupt
23     NVIC_SetPriority(HTCFG_SPI_MASTER_IRQn, 0);
24     NVIC_EnableIRQ(HTCFG_SPI_MASTER_IRQn);
25
26     /* Enable master later
27     SPI_Cmd(HTCFG_SPI_MASTER, ENABLE);
28 }
29
```

Remove SLAVE settings 3

2

SPI_DATALENGTH_16

2400

1

```
45 /* Private constants -----
46 #define BufferSize 1
47 #define Temp_Int_Mask 0xFF
48 #define Temp_Dec_Mask 0x0F
49 #define TC77_LSB 0.0625f
50
51 /* Private function prototypes -----
52 void SPI_Configuration(void);
53 float Convert_Temp(void);
54
55 /* Private variables -----
56 ul6 SPI0_Buffer_Rx[BufferSize] = {0};
57
```

```
58 /* Global functions -----
59 /* *****
60  * @brief Main program.
61  * @retval None
62  * *****
63 int main(void)
64 {
65     RETARGET_Configuration();
66
67     SPI_Configuration();
68
69     while (1){
70         printf("SPI0_Buffer_Rx[0] = 0x%04X / Temperature = %.4f\r\n", SPI0_Buffer_Rx[0], Convert_Temp());
71     }
72 }
73
```

4

```
130 /* *****
131  * @brief Convert the temperature value to real temperature.
132  * @retval None
133  * *****
134 float Convert_Temp(void)
135 {
136     ul6 temp_buffer = SPI0_Buffer_Rx[0];
137     int temp_pol, temp_int, temp_dec;
138     temp_pol = (temp_buffer >> 15) ? -1 : 1;
139     temp_int = (temp_buffer >> 07) & Temp_Int_Mask;
140     temp_dec = (temp_buffer >> 03) & Temp_Dec_Mask;
141     if (temp_pol == -1){
142         temp_int = Temp_Int_Mask - temp_int;
143         temp_dec = Temp_Dec_Mask - temp_dec + 1;
144     }
145     return temp_pol * (temp_int + temp_dec * TC77_LSB);
146 }
```


ht32f5xxx_01_it.c

1

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

Use array defined in main.c

```
116  /**
117   * @brief This function handles SPI Master interrupt.
118   * @retval None
119   */
120  void HTCFG_SPI_MASTER_IRQHandler(void)
121  {
122      if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_RXBNE))
123      {
124          /* Read received data
125          SPI0_Buffer_Rx[0] = SPI_ReceiveData(HTCFG_SPI_MASTER);
126      }
127
128      if (SPI_GetFlagStatus(HTCFG_SPI_MASTER, SPI_FLAG_TXBE))
129      {
130          /* Send data
131          SPI_SendData(HTCFG_SPI_MASTER, 0x00);
132      }
133  }
134  */
```

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

```
139  void HTCFG_SPI_SLAVE_IRQHandler(void)
140  {
141  }
```

Remove content in HTCFG_SPI_SLAVE_IRQHandler

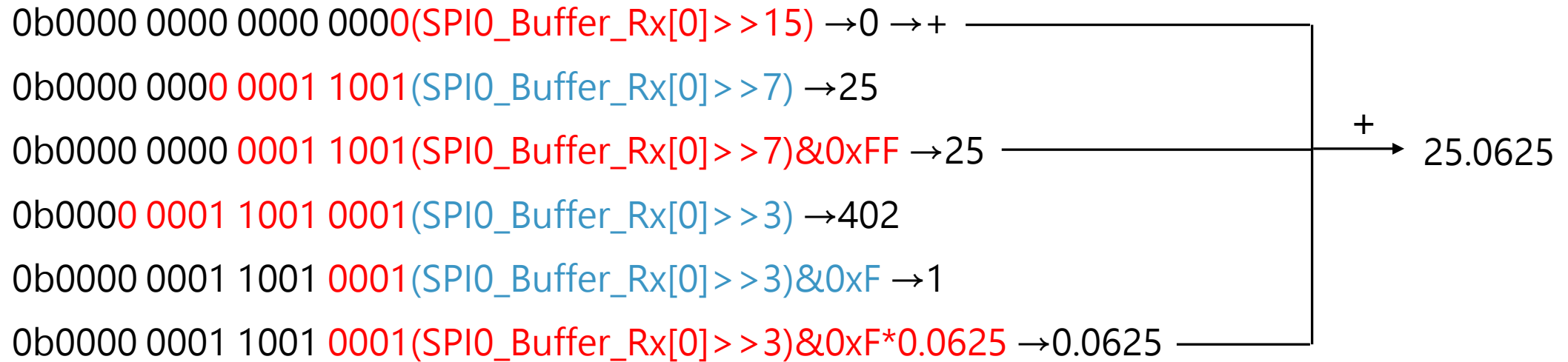
Temperature conversion formula

The data read from TC77: SPI0_Buffer_Rx[0]

For example: 25.0625 °C

0x0C8F(TC77_hexadecimal)

0b0000 1100 1000 1111(TC77_binary)





Class
Dismissed