Experiment 4: Inter-Integrated Circuit I^{2}C

In this experiment, students will learn how to use Inter-Integrated Circuit or I^{2}C module inside STM32F103C8T6. Student will learn how to program Analog to Digital Converter (ADC) module and sent it to computer to display.

1. Inter-Integrated Circuit Module

Inter-Integrated Circuit or I^{2}C is a synchronous serial communication that allows the CPU and connected devices to interact synchronously using the same SDA signal line (channel) and SCL clock signal. By referring to the device, it can be utilized as a multi-master or a master-slave. For each device in the bus system, pass a unique identification number (ID). This protocol was developed by NXP semiconductor, aims to allow ICs or modules to communicate using two signal lines, which are the SDA (Serial Data Line) and SCL (Serial Clock Line). The transmitter will regulate the SCL signal in order to synchronize the connected devices. A unique identifier for the reference device can be used to access any device linked to this I2C bus, allowing identical devices to be attached to the same bus without conflict. The signal SDA and SCL are bi-directional. (Bi-directional) and must be connected to the bus system via an external pull-up resistance in order to function properly. In standard mode (Sm), this system can communicate at 100 kbits/sec, and in high speed communication (Fm) mode, it can communicate at 400 kbits/sec. In I^{2}C, there are two types of device references: 7-bit and 10-bit depends on devices. This protocol is used in master-slave I^{2}C communication. Microprocessor (master) regulates the SCL's timing when reading and writing data to the slaves (devices). In KU Lab board, there are two I^{2}C devices which are PCF8574 and PCF8591. PCF8574 is a remote 8-bit I/O expander while PCF8591 is an 8-bit A/D and D/A converter. These two devices are connected to I^{2}C1 module in STM32F104C8T6. In experiment 4.1, the address of each devices under I^{2}C1 bus on KU Lab board will be searched and displayed.

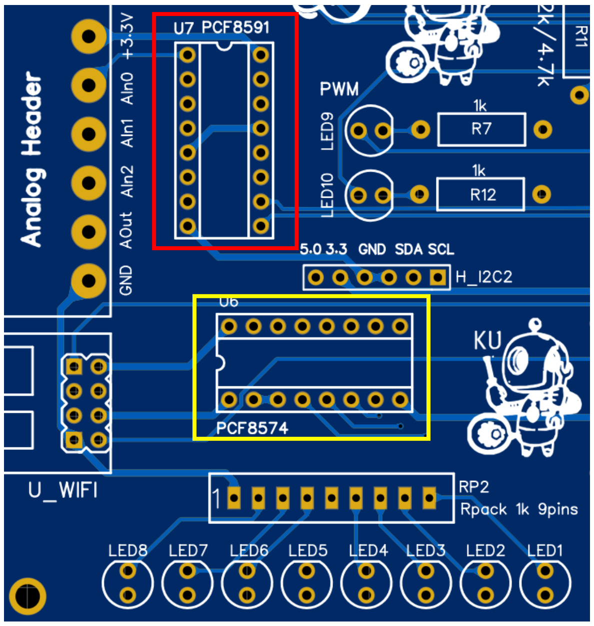


Figure 1. I^{2}C devices on KU lab Board

Experiment 4.1

1. First, new project according to experiment 0 and assign project name as Lab4.
2. Write down the **DelayMC( )** function to the **main.c** file in the project before main( ) function as below:

```c
void DelayMC(unsigned int ms)
{
          volatile unsigned int nCount;
          nCount = (unsigned int)((72000000/4000000)*ms*1000 - 10);
          while(nCount--);
}
```

3. Write down the **GPIO_configuration** function to the **main.c** file in the project before main( ) function as below:

```c
void GPIO_Configuration()
{
   GPIO_InitTypeDef GPIO_InitStruct;
   // Enable clock for GPIOA and GPIOB and AFIO and USART1
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_USART1,ENABLE);
   // Configure PA9 as alternative output push pull for Tx
   GPIO_InitStruct.GPIO_Pin = GPIO_Pin_9;
   GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
   GPIO_Init(GPIOA, &GPIO_InitStruct);
   // Configure PA10 as input floating for Rx
```

```c
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStruct);

    /* I2C1 SDA and SCL configuration */
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_OD;
    GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

4. Add **USART_setup( )**, **USART1_sendC( )** and **USART1_putS( )** functions from **Experiment 2** into the **main.c**. Add the **Initial_I2C1( )**, **__reverse ( )**, **itoa( )** and **SCAN_I2C1( )** functions in **main.c** as follow:

```c
void Initial_I2C1()
{
    I2C_InitTypeDef i2c;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
    I2C_DeInit(I2C1);
    i2c.I2C_ClockSpeed = 100000;
    i2c.I2C_Mode = I2C_Mode_I2C;
    i2c.I2C_DutyCycle = I2C_DutyCycle_2;
    i2c.I2C_OwnAddress1 = 0x4E;
    i2c.I2C_Ack = I2C_Ack_Enable;
    i2c.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
    I2C_Cmd(I2C1, ENABLE);
    I2C_Init(I2C1, &i2c);
}
```

```c
static void __reverse(char* start, char* end)
{
    char tmp;
    while (end > start)
    {
        tmp = *end;
        *end-- = *start;
        *start++ = tmp;
    }
}
```

```c
char* itoa(int value, char* buffer, int base)
{
    static const char digits [] = "0123456789ABCDEF";
    char* tmpBuffer = buffer;
    int sign = 0;
    int quot, rem;
    if ((sign = value) < 0)      /* record sign */
        value = -value;          /* make value positive */

    if ( (base >= 2) && (base <= 16) )
    {
        do
        {   quot = value / base;
            rem = value % base;
            *buffer ++ = digits[rem];
        } while ((value = quot));

    if (sign < 0)
        *buffer ++ = '-';
    __reverse(tmpBuffer, buffer - 1);
    }
*buffer = '\0';
return tmpBuffer;
}
```

```
void SCAN_I2C1()
{
  unsigned char i =0;
  unsigned int timeout = 100000;
  USART1_putS(" <I2C 1 Searching Loop> \n\r");
  unsigned char buf[20];
  int a=0;
  for ( i=5;i<128;i++)
  {
      USART1_putS(" Scaning at Address 0x");
      itoa(i,buf,16);
      USART1_putS(buf);
      USART1_putS("\n\r");
      I2C_ClearFlag(I2C1,I2C_FLAG_AF);
      I2C_AcknowledgeConfig(I2C1,ENABLE);
      I2C_GenerateSTART(I2C1, ENABLE);
      while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
      I2C_ClearFlag(I2C1, I2C_FLAG_AF);

      I2C_Send7bitAddress(I2C1, i <<1 |0, I2C_Direction_Transmitter);
      // check if slave acknowledged his address within timeout
      while((!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED))
&& timeout )
      {
          timeout--;
      }
      DelayMC(50);
      if(I2C_GetFlagStatus(I2C1, I2C_FLAG_AF) != 0x0 );
      if(timeout == 0x0 )
      {  I2C_GenerateSTOP(I2C1, ENABLE);  }
      else
      {
        USART1_putS(">>>>>>>>> Found device at 0x");
        itoa(i,buf,16);
        USART1_putS(buf);
        USART1_putS(" <<<<<<<<<<<\n\r");
        I2C_GenerateSTOP(I2C1, ENABLE);
      }
      DelayMC(50);
      timeout = 100000;
    }
  USART1_putS("<I2C 1 Search End> \n\r");
}
```

5. Edit the **main( )** function in **main.c** as follow:

```
int main(void)
{
  GPIO_Configuration();
  USART_setup();
  Initial_I2C1();
  DelayMC(100);
  SCAN_I2C1();
  while(1);
}
```

6. Compile and program into the board. Open ProcessingGrapher program, set to com port number and Baud rate to 115200 and click connect. Reset the STM32F103C8T6 and find out what is the address for devices that connected to I^{2}C 1 module in STM32F103C8T6.

____

____

____

Lab instructor / TA signature

2. PCF8574

In this topic, 8-bit I/O expander PCF8574 is experimented. The PCF8574 provides general-purpose remote I/O expansion via the two-wire bidirectional I^{2}C-bus. This IC operate at 100 kHz I^{2}C-bus interface (Standard-mode I^{2}C-bus). Pin's name P0 – P7 are the I/O ports from PCF8574. A0 - A2 pins are adjustable address pins as shown in Figure

2. To use this IC, the unique address is set by applying logic to A0 – A2 pins. For KU Lab board, the P0 – P7 pins are connected to LEDs. Since the PCF8574 address is the 7bits address and the least significant bit is used to select Read / Write command as shown in Figure 3. Therefore, to communicate with PCF8574, the address that was found by the experiment 4.1 is needed to be shifted to left by 1 bit and replace the least significant bit to zero.

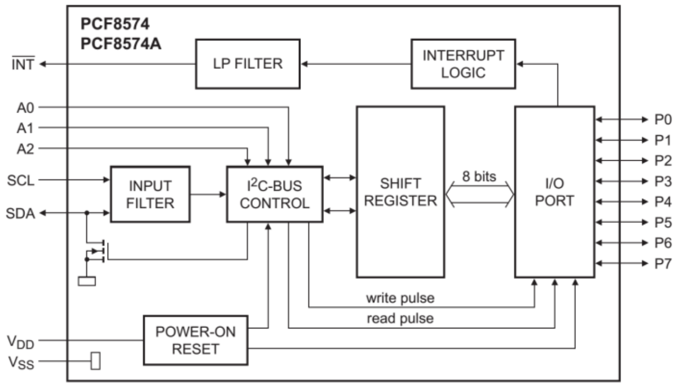


Figure 2. PCF8574 diagram.

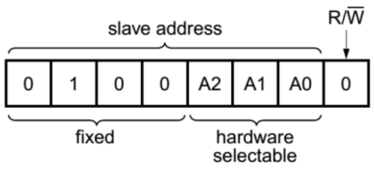


Figure 3. PCF8574 Address.

Experiment 4.2

1. Use the same project as experiment 4.1. Add **write_I2C1( )** function into the **main.c** as below:

```
void write_I2C1(uint8_t addr, uint8_t data)
{
	I2C_ClearFlag(I2C1,I2C_FLAG_AF);
	while(I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY));
	I2C_GenerateSTART(I2C1, ENABLE);
	while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
	I2C_Send7bitAddress(I2C1, addr, I2C_Direction_Transmitter);
	while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
	while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTING));
	I2C_SendData(I2C1, data);
	while(!I2C_GetFlagStatus(I2C1, I2C_FLAG_BTF));
	I2C_GenerateSTOP(I2C1, ENABLE);
	while(I2C_GetFlagStatus(I2C1, I2C_FLAG_STOPF));
}
```

2. Edit the **main( )** function in **main.c** as follow:

```c
int main(void)
{
  unsigned int i = 0;
  GPIO_Configuration();
  Initial_I2C1();
  USART_setup();
  DelayMC(100);
  while(1)
  {
     write_I2C1( PCF8574 address from experiment 4.1 <<1 |0 , i);
     i++;
     DelayMC(80);
  }
}
```

3. Compile and program into the board. Reset the STM32F103C8T6 and explain the result.

____

____

____

____

____

____

____

Lab instructor / TA signature

3. PCF8591

The PCF8591 is a single-chip, single-supply low-power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I^2C-bus interface. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I^2C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I^2C-bus. To communicate with this IC, the address that found from the experiment 4.1 is required to be shifted like PCF8574 case.

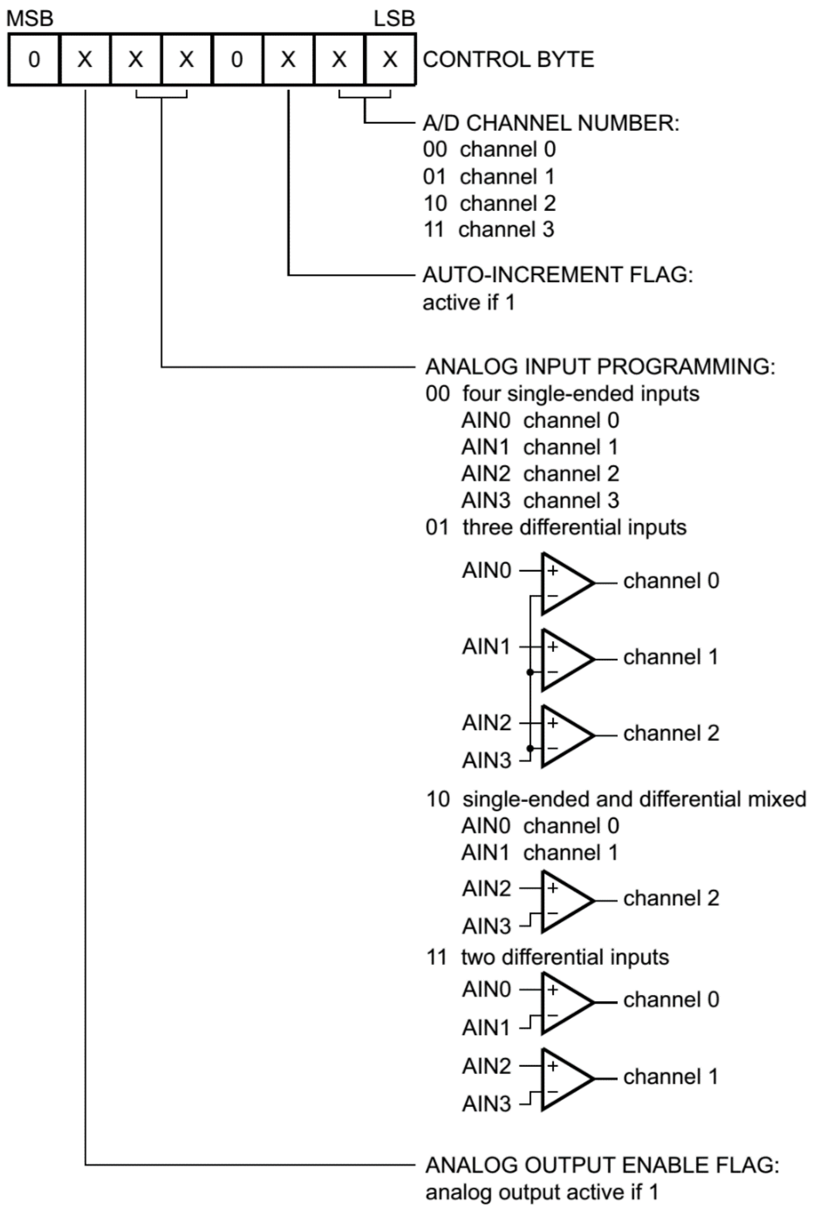


Figure 4. PCF8591 control word.

This ADC IC is more complicated than PCF8574, thus, the control word is need to set the mode of PCF8591 according to Figure 4.

Experiment 4.3

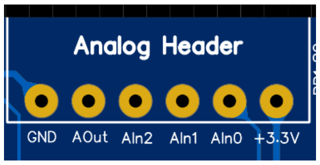
1. Use the same project as experiment 4.2. Add **read_I2C1( )** function into the **main.c** as below:

```c
uint8_t read_I2C1(uint8_t addr)
{
    uint8_t i2c_data;
    I2C_AcknowledgeConfig(I2C1, ENABLE);
    while(I2C_GetFlagStatus(I2C1,I2C_FLAG_BUSY));
    I2C_GenerateSTART(I2C1, ENABLE);
    while(!I2C_GetFlagStatus(I2C1,I2C_FLAG_SB));
    I2C_Send7bitAddress(I2C1,  addr, I2C_Direction_Receiver);
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    i2c_data = I2C_ReceiveData(I2C1);
    I2C_NACKPositionConfig(I2C1, I2C_NACKPosition_Current);
    I2C_AcknowledgeConfig(I2C1, DISABLE);
    I2C_GenerateSTOP(I2C1, ENABLE);
    while(I2C_GetFlagStatus(I2C1, I2C_FLAG_STOPF));
    return i2c_data;
}
```

2. Edit the **main( )** function in **main.c** as follow:

```c
int main(void)
{
  unsigned int i = 0, ADCRead_value = 0;
  unsigned int highV =0 , lowV =0;
  unsigned char sbuf[20];
  GPIO_Configuration();
  Initial_I2C1();
  USART_setup();
  DelayMC(100);
  //Send Control word to PCF8591 //
  write_I2C1( PCF8591 address from experiment 4.1 <<1 |0, 0x00);
  while(1)
  {

     i--;
     ADCRead_value = read_I2C1( PCF8591 address from experiment 4.1 <<1 |0);
     highV = (int)(ADCRead_value*3.3)/255;
     lowV = (int)((ADCRead_value*100/255) * 3.3) % 100;
     itoa(highV, sbuf, 10);
     USART1_putS(sbuf);
     USART1_putS(".");
     itoa(lowV, sbuf, 10);
     USART1_putS(sbuf);
     USART1_putS("\n\r");
     DelayMC(80);
  }
}
```

3. Compile and program into the board. Connect variable resister to Analog Header at location +3.3V, Ain0 and GND as shown in Figure 5.



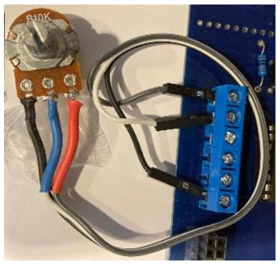


Figure 5. Aout pin.

Connect the USB-to-Serial to desktop or notebook and find the virtual com port number via device manager. Open ProcessingGrapher program, set to com port number and Baud rate to 115200 and click connect. Reset the STM32F103C8T6 and explain the result.

____

____

____

____

____

Lab instructor / TA signature

4. Digital to Analog

Since the PCF8591 has one analog output, thus, in this topic the Digital to Analog or DAC is used. DAC mode can be set using the control word as shown in Figure 6.

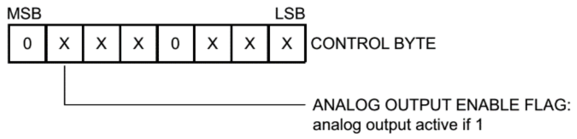


Figure 6. Analog output control word

Experiment 4.3

1. Use the same project as experiment 4.3. Edit the **main( )** function in **main.c** as follow and create the new function **write_dac_I2C1( )** :

```c
void write_dac_I2C1(uint8_t addr, uint8_t data)
{
	I2C_ClearFlag(I2C1,I2C_FLAG_AF);
	while(I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY));
	I2C_GenerateSTART(I2C1, ENABLE);
	while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
	I2C_Send7bitAddress(I2C1, addr, I2C_Direction_Transmitter);
	while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
	while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTING));
	I2C_SendData(I2C1, 0x40);
	while(!I2C_GetFlagStatus(I2C1, I2C_FLAG_BTF));
	I2C_SendData(I2C1, data);
	while(!I2C_GetFlagStatus(I2C1, I2C_FLAG_BTF));
	I2C_GenerateSTOP(I2C1, ENABLE);
	while(I2C_GetFlagStatus(I2C1, I2C_FLAG_STOPF));
}
```

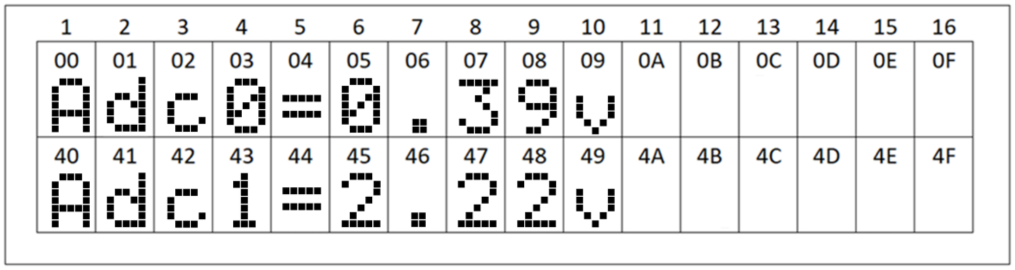
```c
int main(void)
{
  unsigned char i = 0;
  GPIO_Configuration();
  USART_setup();
  Initial_I2C1();
  DelayMC(100);
  //Send Control word to PCF8591 //

  while(1)
  {
     write_dac_I2C1( PCF8591 address from experiment 4.1 <<1 |0 , i);
     i++;
     DelayMC(20);
  }
}
```

2. Compile and program into the board. Use oscilloscope to measure the output at Aout shown in Figure 5 and plot graph and calculate the frequency of the output signal.

Output frequency = _________________ Hz

Homework

1. Combine LCD module code from experiment 1 with I^2C ADC read code in this experiment, then, shows the ADC channel 0 and 1 values on LCD screen as follow:


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
Adc0=0.39v
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
Adc1=2.22v

2. Use I^2C DAC to generate output signal $y = 0.8\sin(2\pi f t) + 1.0$ v, when f can be selected by yourself.

Lab instructor / TA signature