# 7 STARTING with SERIAL COMMUNICATION – Code Snips

**Note: These Code Snips are taken straight from the book chapter; i.e. the “Program Examples”. In some cases therefore they are not complete programs.**

/\* Program Example 7.1: Sets up the mbed as SPI Master, and continuously sends

a single byte

\*/

#include "mbed.h"

SPI ser\_port(p11, p12, p13); // mosi, miso, sclk

char switch\_word ; //word we will send

int main() {

ser\_port.format(8,0); // Setup the SPI for 8 bit data, Mode 0 operation

ser\_port.frequency(1000000); // Clock frequency is 1MHz

while (1){

switch\_word=0xA1; //set up word to be transmitted

ser\_port.write(switch\_word); //send switch\_word

wait\_us(50);

}

}

Program Example 7.1: Minimal SPI Master application

/\*Program Example 7.2. Sets the mbed up as Master, and exchanges data with a Slave, sending its own switch positions, and displaying those of the Slave.

\*/

#include "mbed.h"

SPI ser\_port(p11, p12, p13); // mosi, miso, sclk

DigitalOut red\_led(p25); //red led

DigitalOut green\_led(p26); //green led

DigitalOut cs(p14); //this acts as “Slave select”

DigitalIn switch\_ip1(p5);

DigitalIn switch\_ip2(p6);

char switch\_word ; //word we will send

char recd\_val; //value return from Slave

int main() {

while (1){

//Default settings for SPI Master chosen, no need for further configuration

//Set up the word to be sent, by testing switch inputs

switch\_word=0xa0; //set up a recognisable output pattern

if (switch\_ip1==1)

switch\_word=switch\_word|0x01; //OR in lsb

if (switch\_ip2==1)

switch\_word=switch\_word|0x02; //OR in next lsb

cs = 0; //select Slave

recd\_val=ser\_port.write(switch\_word); //send switch\_word and receive data

cs = 1;

wait(0.01);

//set leds according to incoming word from Slave

red\_led=0; //preset both to 0

green\_led=0;

recd\_val=recd\_val&0x03; //AND out unwanted bits

if (recd\_val==1)

red\_led=1;

if (recd\_val==2)

green\_led=1;

if (recd\_val==3){

red\_led=1;

green\_led=1;

}

}

}

Program Example 7.2: The mbed set up as SPI Master, with bidirectional data transfer

/\*Program Example 7.3: Sets the mbed up as Slave, and exchanges data with a Master, sending its own switch positions, and displaying those of the Master. as SPI Slave.

\*/

#include "mbed.h"

SPISlave ser\_port(p11,p12,p13,p14); // mosi, miso, sclk, ssel

DigitalOut red\_led(p25); //red led

DigitalOut green\_led(p26); //green led

DigitalIn switch\_ip1(p5);

DigitalIn switch\_ip2(p6);

char switch\_word ; //word we will send

char recd\_val; //value received from Master

int main() {

//default formatting applied

while(1) {

//set up switch\_word from switches that are pressed

switch\_word=0xa0; //set up a recognisable output pattern

if (switch\_ip1==1)

switch\_word=switch\_word|0x01;

if (switch\_ip2==1)

switch\_word=switch\_word|0x02;

if(ser\_port.receive()) { //test if data transfer has occurred

recd\_val = ser\_port.read(); // Read byte from Master

ser\_port.reply(switch\_word); // Make this the next reply

}

//now set leds according to received word

...

(continues as in Program Example 7.2)

...

}

}

Program Example 7.3: The mbed set up as SPI Slave, with bidirectional data transfer

/\*Program Example 7.4: Reads values from accelerometer through SPI, and outputs

continuously to terminal screen.

\*/

#include "mbed.h“

SPI acc(p11,p12,p13); // set up SPI interface on pins 11,12,13

DigitalOut cs(p14); // use pin 14 as chip select

Serial pc(USBTX, USBRX); // set up USB interface to host terminal

char buffer[6]; // raw data array type char

int16\_t data[3]; // 16-bit twos-complement integer data

float x, y, z; // floating point data, to be displayed on-screen

int main() {

cs=1; // initially ADXL345 is not activated

acc.format(8,3); // 8 bit data, Mode 3

acc.frequency(2000000); // 2MHz clock rate

cs=0; // select the device

acc.write(0x31); // data format register

acc.write(0x0B); // format +/-16g, 0.004g/LSB

cs=1; // end of transmission

cs=0; // start a new transmission

acc.write(0x2D); // power ctrl register

acc.write(0x08); // measure mode

cs=1; // end of transmission

while (1) { // infinite loop

wait(0.2);

cs=0; // start a transmission

acc.write(0x80|0x40|0x32); // RW bit high, MB bit high, plus address

for (int i = 0;i<=5;i++) {

buffer[i]=acc.write(0x00); // read back 6 data bytes

}

cs=1; // end of transmission

data[0] = buffer[1]<<8 | buffer[0]; // combine MSB and LSB

data[1] = buffer[3]<<8 | buffer[2];

data[2] = buffer[5]<<8 | buffer[4];

x=0.004\*data[0]; y=0.004\*data[1]; z=0.004\*data[2]; // convert to float,

// actual g value

pc.printf("x = %+1.2fg\t y = %+1.2fg\t z = %+1.2fg\n\r", x, y,z); // print

}

}

Program Example 7.4: Accelerometer continuously outputs 3-axis data to terminal screen

/\*Program Example 7.5: I2C Master, transfers switch state to second mbed acting as Slave, and displays state of Slave's switches on its leds.

\*/

#include "mbed.h"

I2C i2c\_port(p9, p10); //Configure a serial port, pins 9 and 10 are sda, scl

DigitalOut red\_led(p25); //red led

DigitalOut green\_led(p26); //green led

DigitalIn switch\_ip1(p5); //input switch

DigitalIn switch\_ip2(p6);

char switch\_word ; //word we will send

char recd\_val; //value received from Slave

const int addr = 0x52; //the I2C Slave address, an arbitrary even number

int main() {

while(1) {

switch\_word=0xa0; //set up a recognisable output pattern

if (switch\_ip1==1)

switch\_word=switch\_word|0x01; //OR in lsb

if (switch\_ip2==1)

switch\_word=switch\_word|0x02; //OR in next lsb

//send a single byte of data, in correct I2C package

i2c\_port.start(); //force a start condition

i2c\_port.write(addr); //send the address

i2c\_port.write(switch\_word); //send one byte of data, ie switch\_word

i2c\_port.stop(); //force a stop condition

wait(0.002);

//receive a single byte of data, in correct I2C package

i2c\_port.start();

i2c\_port.write(addr|0x01); //send address, with R/W bit set to Read

recd\_val=i2c\_port.read(addr); //Read and save the received byte

i2c\_port.stop(); //force a stop condition

//set leds according to word received from Slave

red\_led=0; //preset both to 0

...

(continues as in Program Example 7.2)

...

}

}

Program Example 7.5: I2C data link Master

/\*Program Example 7.6: I2C Slave, when called transfers switch state to mbed acting as Master, and displays state of Master's switches on its leds.

\*/

#include <mbed.h>

I2CSlave slave(p9, p10); //Configure I2C Slave

DigitalOut red\_led(p25); //red led

DigitalOut green\_led(p26); //green led

DigitalIn switch\_ip1(p5);

DigitalIn switch\_ip2(p6);

char switch\_word ; //word we will send

char recd\_val; //value received from Master

int main() {

slave.address(0x52);

while (1) {

//set up switch\_word from switches that are pressed

switch\_word=0xa0; //set up a recognisable output pattern

if (switch\_ip1==1)

switch\_word=switch\_word|0x01;

if (switch\_ip2==1)

switch\_word=switch\_word|0x02;

slave.write(switch\_word); //load up word to send

//test for I2C, and act accordingly

int i = slave.receive();

if (i == 3){ //Slave is addressed, Master will write

recd\_val= slave.read();

//now set leds according to received word

...

(continues as in Program Example 7.2)

...

}

}

Program Example 7.6: I2C data link Slave

/\*Program Example 7.7: Mbed communicates with TMP102 temperature sensor, and scales and displays readings to screen.

\*/

#include "mbed.h“

I2C tempsensor(p9, p10); //sda, sc1

Serial pc(USBTX, USBRX); //tx, rx

const int addr = 0x90;

char config\_t[3];

char temp\_read[2];

float temp;

int main() {

config\_t[0] = 0x01; //set pointer reg to 'config register'

config\_t[1] = 0x60; // config data byte1

config\_t[2] = 0xA0; // config data byte2

tempsensor.write(addr, config\_t, 3);

config\_t[0] = 0x00; //set pointer reg to 'data register'

tempsensor.write(addr, config\_t, 1); //send to pointer 'read temp'

while(1) {

wait(1);

tempsensor.read(addr, temp\_read, 2); //read the two-byte temp data

temp = 0.0625 \* (((temp\_read[0] << 8) + temp\_read[1]) >> 4); //convert data

pc.printf("Temp = %.2f degC\n\r", temp);

}

}

Program Example 7.7: Communicating by I2C with the TMP102 temperature sensor

/\*Program Example 7.8: Configures and takes readings from the SRF08 ultrasonic range finder, and displays them on screen.

\*/

#include "mbed.h"

I2C rangefinder(p9, p10); //sda, sc1

Serial pc(USBTX, USBRX); //tx, rx

const int addr = 0xE0;

char config\_r[2];

char range\_read[2];

float range;

int main() {

while (1) {

config\_r[0] = 0x00; //set pointer reg to ‘cmd register'

config\_r[1] = 0x51; //initialise, result in cm

rangefinder.write(addr, config\_r, 2);

wait(0.07);

config\_r[0] = 0x02; //set pointer reg to 'data register'

rangefinder.write(addr, config\_r, 1); //send to pointer 'read range'

rangefinder.read(addr, range\_read, 2); //read the two-byte range data

range = ((range\_read[0] << 8) + range\_read[1]);

pc.printf("Range = %.2f cm\n\r", range); //print range on screen

wait(0.05);

}

}

Program Example 7.8: Communicating by I2C with the SRF08 range finder

/\*Program Example 7.9: Sets the mbed up for async communication, and exchanges data with a similar node, sending its own switch positions, and displaying those of the other.

\*/

#include "mbed.h"

Serial async\_port(p9, p10); //set up TX and RX on pins 9 and 10

DigitalOut red\_led(p25); //red led

DigitalOut green\_led(p26); //green led

DigitalOut strobe(p7); //a strobe to trigger the scope

DigitalIn switch\_ip1(p5);

DigitalIn switch\_ip2(p6);

char switch\_word ; //the word we will send

char recd\_val; //the received value

int main() {

async\_port.baud(9600); //set baud rate to 9600 (ie default)

//accept default format, of 8 bits, no parity

while (1){

//Set up the word to be sent, by testing switch inputs

switch\_word=0xa0; //set up a recognisable output pattern

if (switch\_ip1==1)

switch\_word=switch\_word|0x01; //OR in lsb

if (switch\_ip2==1)

switch\_word=switch\_word|0x02; //OR in next lsb

strobe =1; //short strobe pulse

wait\_us(10);

strobe=0;

async\_port.putc(switch\_word); //transmit switch\_word

if (async\_port.readable()==1) //is there a character to be read?

recd\_val=async\_port.getc(); //if yes, then read it

...

(continues as in Program Example 7.2)

...

}

}

**Program Example 7.9: Bidirectional data transfer between two mbed UARTs**

/\* Program Example 7.10: Emulating a USB mouse

\*/

#include "mbed.h" // include mbed library

#include "USBMouse.h" // include USB Mouse library

USBMouse mouse; // define USBMouse interface

int dx[]={40,0,-40,0}; // relative x position co-ordinates

int dy[]={0,40,0,-40}; // relative y position co-ordinates

int main() {

while (1) {

for (int i=0; i<4; i++) { // scroll through position co-ordinates

mouse.move(dx[i],dy[i]); // move mouse to co-ordinate

wait(0.2);

}

}

}

Program Example 7.10: Emulating a USB mouse