SPI Programming

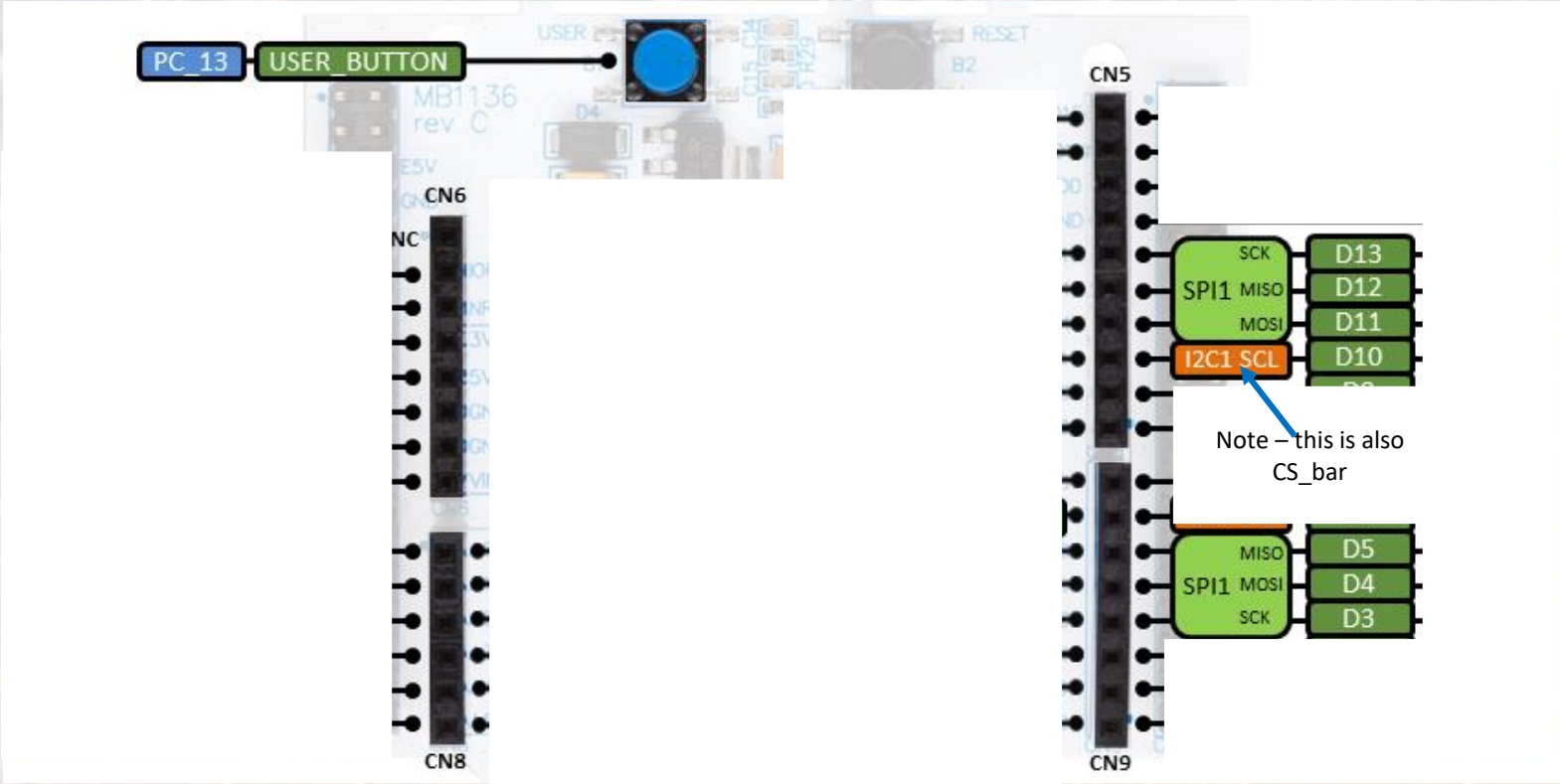
Last updated 6/14/21

SPI Programming

- SPI Operation
  - Nucleo-L476RG has 3 SPI modules
  - Only one of these available on the Arduino headers
  - SPI1 and SPI2 have two sets of access ports
  - Mbed defaults to MSB first transfer only
  - Master uses SPI class
  - Slave uses SPISlave class

SPI Programming

- SPI Connections
  - Arduino



SPI Programming

- SPI Connections
  - Morpho

SPI Programming (master)

- SPI Class

Public Member Functions	
	SPI (PinName mosi, PinName miso, PinName sclk, PinName ssel=NC)
	Create a SPI master connected to the specified pins. More...
	SPI (PinName mosi, PinName miso, PinName sclk, PinName ssel, use_gpio_ssel_t)
	Create a SPI master connected to the specified pins. More...
	SPI (const spi_pinmap_t &static_pinmap)
	Create a SPI master connected to the specified pins. More...
	SPI (const spi_pinmap_t &static_pinmap, PinName ssel)
	Create a SPI master connected to the specified pins. More...
void	format (int bits, int mode=0)
	Configure the data transmission format. More...
void	frequency (int hz=1000000)
	Set the SPI bus clock frequency. More...
virtual int	write (int value)
	Write to the SPI Slave and return the response. More...
virtual int	write (const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length)
	Write to the SPI Slave and obtain the response. More...
virtual void	lock (void)
	Acquire exclusive access to this SPI bus. More...
virtual void	unlock (void)
	Release exclusive access to this SPI bus. More...

Mode	Polarity	Phase
0	0	0
1	0	1
2	1	0
3	1	1

void	select (void)
	Assert the Slave Select line, acquiring exclusive access to this SPI bus. More...
void	deselect (void)
	Deassert the Slave Select line, releasing exclusive access to this SPI bus. More...
void	set_default_write_value (char data)
	Set default write data. More...
template<typename Type >	
int	transfer (const Type *tx_buffer, int tx_length, Type *rx_buffer, int rx_length, const event_callback_t &callback, int event=SPI_EVENT_COMPLETE)
	Start non-blocking SPI transfer using 8bit buffers. More...
void	abort_transfer ()
	Abort the on-going SPI transfer, and continue with transfers in the queue, if any. More...
void	clear_transfer_buffer ()
	Clear the queue of transfers. More...
void	abort_all_transfers ()
	Clear the queue of transfers and abort the on-going transfer. More...
int	set_dma_usage (DMAUsage usage)
	Configure DMA usage suggestion for non-blocking transfers. More...

SPI Programming (master)

- Constructor

Public Member Functions	
	SPI (PinName mosi, PinName miso, PinName sclk, PinName ssel=NC)
	Create a SPI master connected to the specified pins. More...
	SPI (PinName mosi, PinName miso, PinName sclk, PinName ssel, use_gpio_ssel_t)
	Create a SPI master connected to the specified pins. More...
	SPI (const spi_pinmap_t &static_pinmap)
	Create a SPI master connected to the specified pins. More...
	SPI (const spi_pinmap_t &static_pinmap, PinName ssel)
	Create a SPI master connected to the specified pins. More...

```
// Create and configure the SPI object
// Using tha Master only module - so no SSEL
SPI Spi_tx(D11, D12, D13);  // MOSI, MISO, SCK
```

SPI Programming (master)

- Member Functions (Methods)

void	format (int bits, int mode=0)
	Configure the data transmission format. More...
void	frequency (int hz=1000000)
	Set the SPI bus clock frequency. More...
virtual int	write (int value)
	Write to the SPI Slave and return the response. More...
virtual int	write (const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length)
	Write to the SPI Slave and obtain the response. More...
virtual void	lock (void)
	Acquire exclusive access to this SPI bus. More...
virtual void	unlock (void)
	Release exclusive access to this SPI bus. More...

Mode	Polarity	Phase
0	0	0
1	0	1
2	1	0
3	1	1

void	select (void)
	Assert the Slave Select line, acquiring exclusive access to this SPI bus. More...
void	deselect (void)
	Deassert the Slave Select line, releasing exclusive access to this SPI bus. More...
void	set_default_write_value (char data)
	Set default write data. More...
template<typename Type >	
int	transfer (const Type *tx_buffer, int tx_length, Type *rx_buffer, int rx_length, const event_callback_t &callback, int event=SPI_EVENT_COMPLETE)
	Start non-blocking SPI transfer using 8bit buffers. More...
void	abort_transfer ()
	Abort the on-going SPI transfer, and continue with transfers in the queue, if any. More...
void	clear_transfer_buffer ()
	Clear the queue of transfers. More...
void	abort_all_transfers ()
	Clear the queue of transfers and abort the on-going transfer. More...
int	set_dma_usage (DMAUsage usage)
	Configure DMA usage suggestion for non-blocking transfers. More...

```
// Create and configure the SPI object
// Using tha Master only module - so no SSEL
SPI Spi_tx(D11, D12, D13);  // MOSI, MISO, SCK
Spi_tx.format(8, 0);        // 8 bit transfer, pha=0, pol=0
Spi_tx.frequency(1000000);  // 1MHz baud rate

// loop through consecutive write values
while(1){
    Spi_tx.write(count);
```

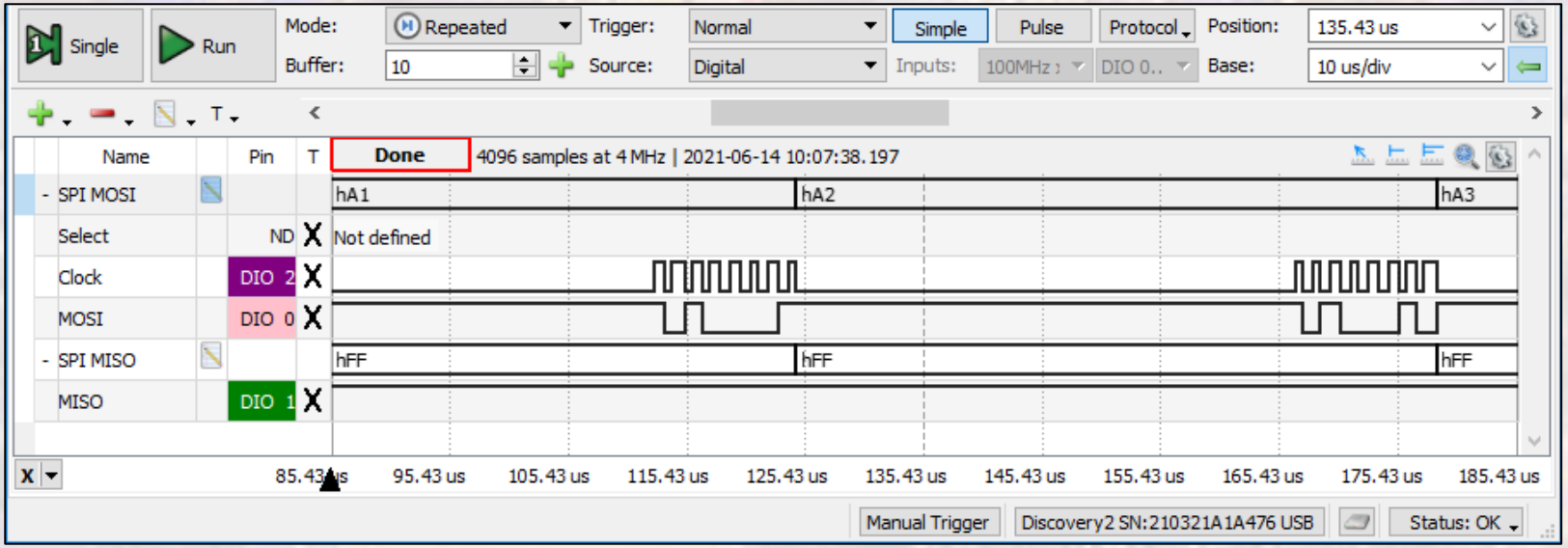
SPI Programming (master)

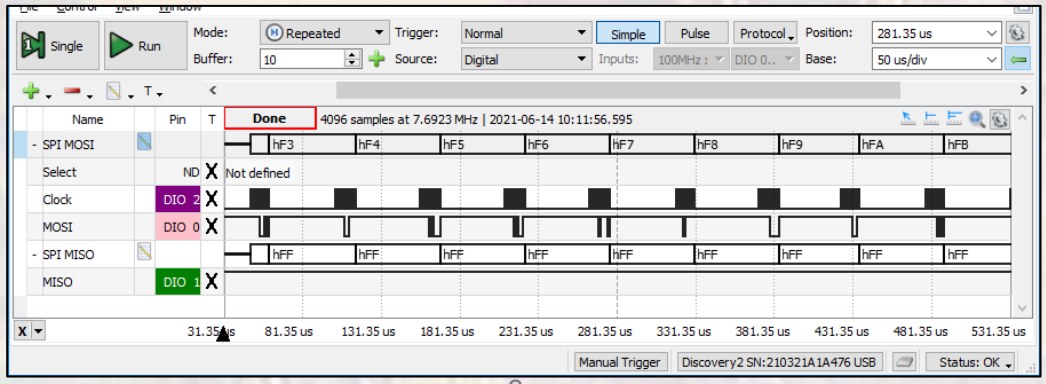
- Simple example 1
  - Transmit a series of counts

```
////////////////////////////////////
//
// spi_class_ex_1 project
//
// created 6/14/21 by tj
// rev 0
//
////////////////////////////////////
//
// SPI example file for class
//
// SPI write only
// uses AD2 to see spi writes
//
////////////////////////////////////

#include "mbed.h"
#include <stdio.h>

// Global HARDWARE Objects
// Create the SPI object
SPI Spi_tx(D11, D12, D13);  // MOSI, MISO, SCK

int main(void){
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("\n\nspi_class_ex_1 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
           MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // working variables
    int count;
    count = 0;

    // Configure the SPI object
    // Using tha Master only module - so no SSEL
    Spi_tx.format(8, 0);        // 8 bit transfer, pha=0, pol=0
    Spi_tx.frequency(1000000);  // 1MHz baud rate

    // loop through consecutive write values
    while(1){
        Spi_tx.write(count);
        wait_us(25);
        count++;
    } // end while

    return 0;
}// end main
```

SPI Programming (master)

- Simple example 1
  - Transmit a series of counts





SPISlave Programming

- SPISlave Class

Public Member Functions	
	SPISlave (PinName mosi, PinName miso, PinName sclk, PinName ssel)
	Create a SPI slave connected to the specified pins. More...
	SPISlave (const spi_pinmap_t &pinmap)
	Create a SPI slave connected to the specified pins. More...
void	format (int bits, int mode=0)
	Configure the data transmission format. More...
void	frequency (int hz=1000000)
	Set the SPI bus clock frequency. More...
int	receive (void)
	Polls the SPI to see if data has been received. More...
int	read (void)
	Retrieve data from receive buffer as slave. More...
void	reply (int value)
	Fill the transmission buffer with the value to be written out as slave on the next received message from the master. More...

Mode	Polarity	Phase
0	0	0
1	0	1
2	1	0
3	1	1

SPISlave Programming

- Constructor

Public Member Functions	
	SPISlave (PinName mosi, PinName miso, PinName sclk, PinName ssel)
	Create a SPI slave connected to the specified pins. More...
	SPISlave (const spi_pinmap_t &pinmap)
	Create a SPI slave connected to the specified pins. More...

```
// Create and configure the RX SPI object
// Using tha Slave only module - SSEL required
SPISlave Spi_S(PB_15, PB_14, PB_13, PB_12);  // MOSI, MISO, SCK, SSEL-not(CS)
```

SPISlave Programming

- Member Functions (Methods)

void	format (int bits, int mode=0)
	Configure the data transmission format. More...
void	frequency (int hz=1000000)
	Set the SPI bus clock frequency. More...
int	receive (void)
	Polls the SPI to see if data has been received. More...
int	read (void)
	Retrieve data from receive buffer as slave. More...
void	reply (int value)
	Fill the transmission buffer with the value to be written out as slave on the next received message from the master. More...

Mode	Polarity	Phase
0	0	0
1	0	1
2	1	0
3	1	1

```
// Create and configure the RX SPI object
// Using tha Slave only module - SSEL required
SPISlave Spi_S(PB_15, PB_14, PB_13, PB_12);  // MOSI, MISO, SCK, SSEL-not(CS)
Spi_S.format(8, 0);          // 8 bit transfer, pha=0, pol=0
Spi_S.frequency(1000000);    // 1MHz baud rate
```

```
// set reply for save
Spi_S.reply(255 - m_tx);
```

```
// read from slave
s_rx = Spi_S.read();
```

SPISlave Programming

- Simple example 2
  - Loopback from master to slave
  - Uses Morpho pins for SPI2 access

```
////////////////////////////////////
//
// spi_class_ex_2 project
//
// created 6/14/21 by tj
// rev 0
//
////////////////////////////////////
//
// SPI example file for class
//
// SPI loop back to show read
//
////////////////////////////////////

#include "mbed.h"
#include <stdio.h>

// Global HARDWARE Objects
// Create the TX SPI object
SPI Spi_M(D4, D5, D3);  // MOSI, MISO, SCK
// Create a digital output to use for SS_bar
DigitalOut Ssel_bar(D8);
// Create the RX SPI object
SPISlave Spi_S(PB_15, PB_14, PB_13, PB_12);  // MOSI, MISO, SCK, SSEL-not(CS)

int main(void){
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("\n\nspi_class_ex_2 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
           MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // working variables
    uint8_t m_tx;
    uint8_t m_rx;
    uint8_t s_rx;
    m_tx = 0;

    // Configure the TX SPI object
    // Using tha Master only module - so no SSEL
    Spi_M.format(8, 0);        // 8 bit transfer, pha=0, pol=0
    Spi_M.frequency(1000000);  // 1MHz baud rate
```

```
    // force SSEL_bar high to program the slave
    Ssel_bar = 1;

    // Configure the RX SPI object
    // Using tha Slave only module - SSEL required
    Spi_S.format(8, 0);        // 8 bit transfer, pha=0, pol=0
    Spi_S.frequency(1000000);  // 1MHz baud rate

    // Activate the slave by taking SSEL_bar low
    Ssel_bar = 0;

    // loop through consecutive write values
    while(1){
        // set reply for save
        Spi_S.reply(255 - m_tx);

        // write from master to slave and save the co-incident response
        m_rx = Spi_M.write(m_tx);

        wait_us(1000);

        // read from slave
        s_rx = Spi_S.read();

        printf("Master tx: %i\t Slave tx: %i\t Master rx: %i\t Slave rx: %i\n", m_tx, (255 - m_tx), m_rx, s_rx);

        wait_us(1000);
        m_tx++;
    } // end while

    return 0;
}// end main
```

SPISlave Programming

- Simple example 2
  - Loopback from master to slave
  - Uses Morpho pins for SPI2 access

```
spi_class_ex_2 - example for EE2905
Using Mbed OS version 6.10.0

Master tx: 0     Slave tx: 255   Master rx: 255  Slave rx: 0
Master tx: 1     Slave tx: 254   Master rx: 254  Slave rx: 1
Master tx: 2     Slave tx: 253   Master rx: 253  Slave rx: 2
Master tx: 3     Slave tx: 252   Master rx: 252  Slave rx: 3
Master tx: 4     Slave tx: 251   Master rx: 251  Slave rx: 4
Master tx: 5     Slave tx: 250   Master rx: 250  Slave rx: 5
Master tx: 6     Slave tx: 249   Master rx: 249  Slave rx: 6
Master tx: 7     Slave tx: 248   Master rx: 248  Slave rx: 7
Master tx: 8     Slave tx: 247   Master rx: 247  Slave rx: 8
Master tx: 9     Slave tx: 246   Master rx: 246  Slave rx: 9
Master tx: 10    Slave tx: 245   Master rx: 245  Slave rx: 10
Master tx: 11    Slave tx: 244   Master rx: 244  Slave rx: 11
Master tx: 12    Slave tx: 243   Master rx: 243  Slave rx: 12
Master tx: 13    Slave tx: 242   Master rx: 242  Slave rx: 13
Master tx: 14    Slave tx: 241   Master rx: 241  Slave rx: 14
Master tx: 15    Slave tx: 240   Master rx: 240  Slave rx: 15
Master tx: 16    Slave tx: 239   Master rx: 239  Slave rx: 16
```

SPI Programming

- Limitations summary
  - It looks like an 8-bit write takes ~25us
  - No read for the Master Rx buffer
    - Must capture the read value when you do a write