I2C Programming

Last updated 7/22/21

I2C Programming

- I2C Operation
  - Nucleo-L476RG has 3 I2C modules
  - 2 available on the Arduino headers
  - No pull-up resistors included – you must add these

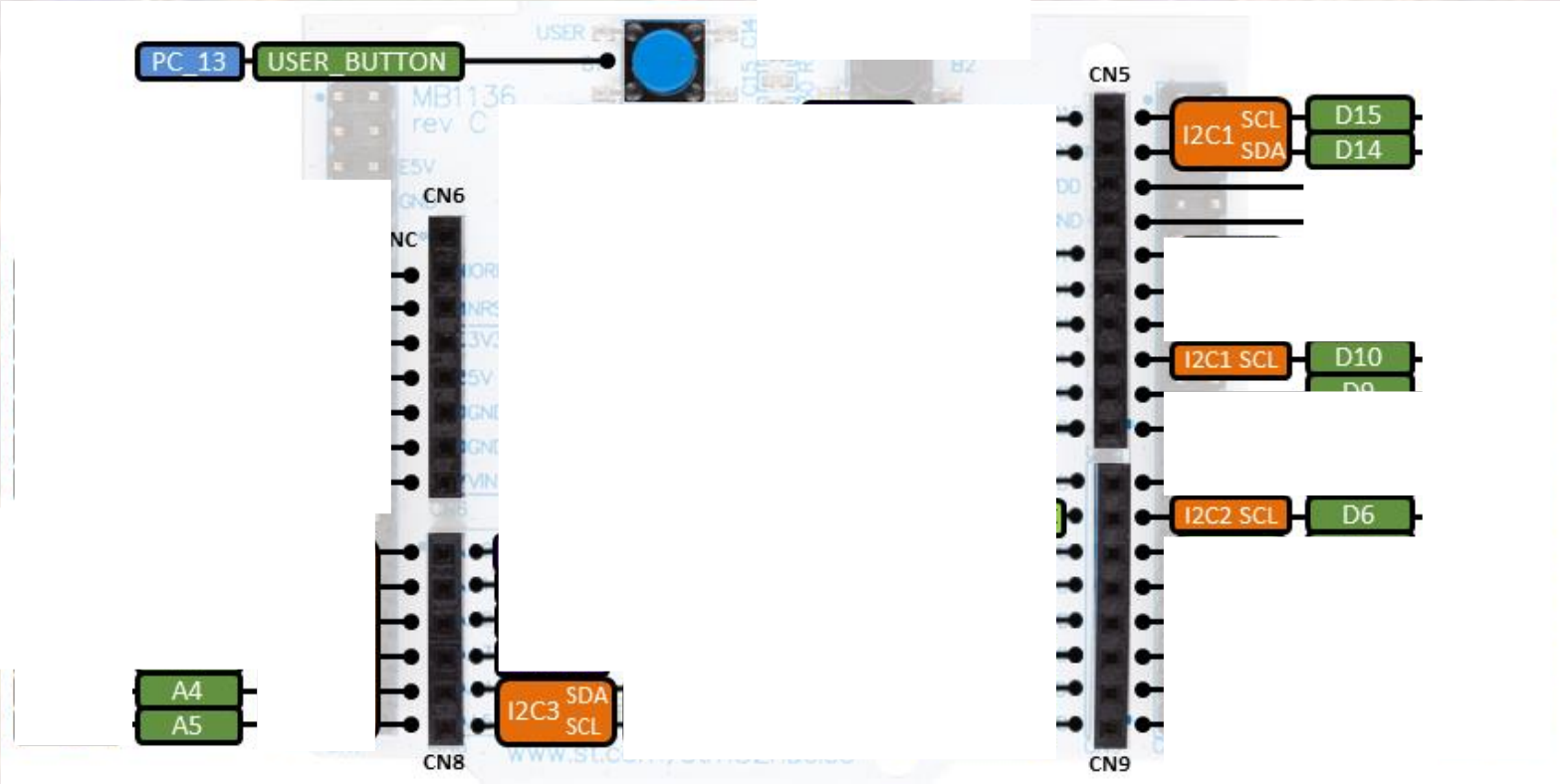
I2C Programming

- Critical Note:
  - The address used in Mbed is the full byte
    - with the rd/wr bit ignored
  - Most systems use just the 7 bits
    - The rd/wr bit is managed separately
  - Address 0x1A for most systems will be 0x34 in Mbed
    or 0x35 in Mbed

x001 1010	0011 0100
x001 1010	0011 0101

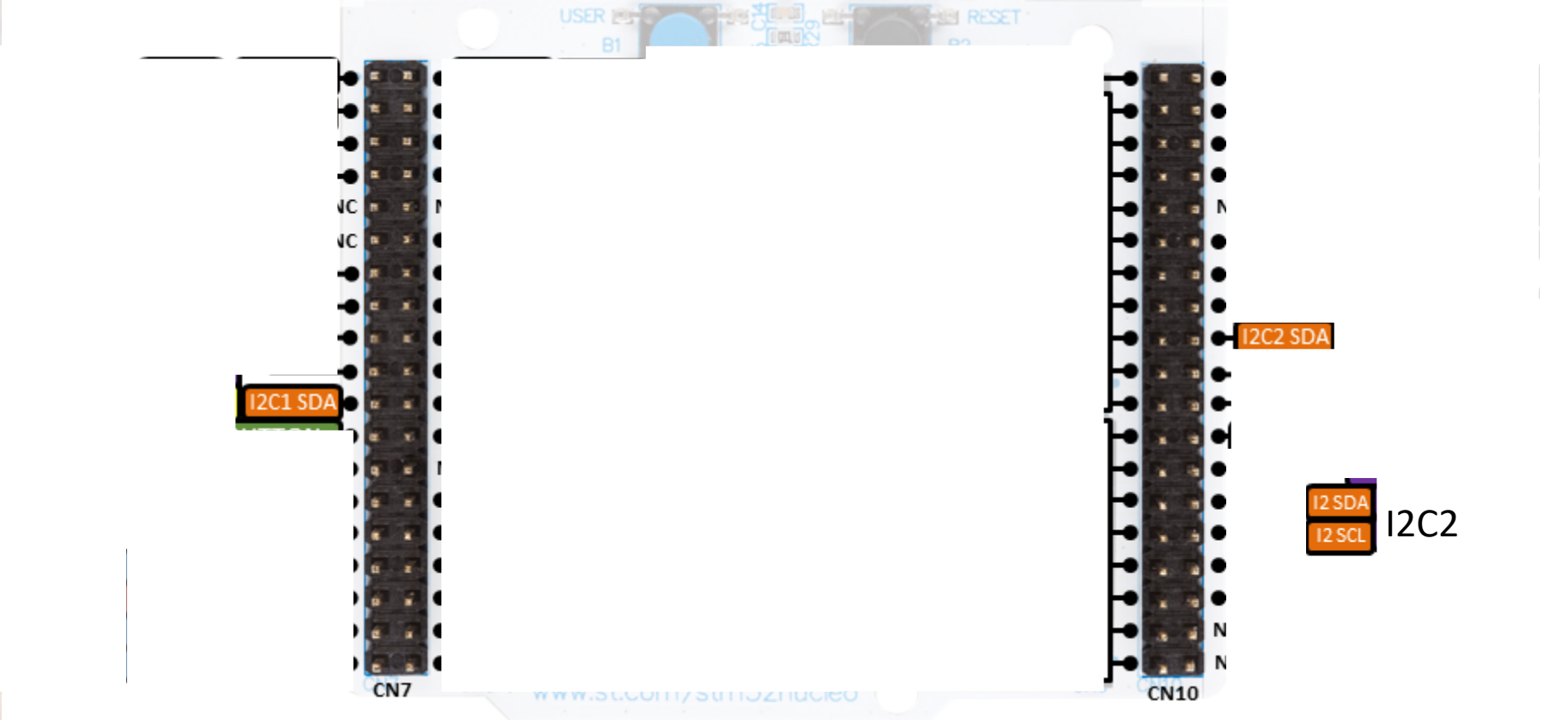
I2C Programming

- I2C Connections
  - Arduino



I2C Programming

- I2C Connections
  - Morpho



I2C Programming

- I2C Master Class

Public Member Functions	
	I2C (PinName sda, PinName scl)
	Create an I2C Master interface, connected to the specified pins. More...
	I2C (const i2c_pinmap_t &static_pinmap)
	Create an I2C Master interface, connected to the specified pins. More...
void	frequency (int hz)
	Set the frequency of the I2C interface. More...
int	read (int address, char *data, int length, bool repeated=false)
	Read from an I2C slave. More...
int	read (int ack)
	Read a single byte from the I2C bus. More...
int	write (int address, const char *data, int length, bool repeated=false)
	Write to an I2C slave. More...
int	write (int data)
	Write single byte out on the I2C bus. More...
void	start (void)
	Creates a start condition on the I2C bus. More...
void	stop (void)
	Creates a stop condition on the I2C bus. More...

virtual void	lock (void)
	Acquire exclusive access to this I2C bus. More...
virtual void	unlock (void)
	Release exclusive access to this I2C bus. More...
int	transfer (int address, const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length, const event_callback_t &callbac event=I2C_EVENT_TRANSFER_COMPLETE, bool repeated=false)
	Start nonblocking I2C transfer. More...
void	abort_transfer ()
	Abort the ongoing I2C transfer. More...

I2C Programming

- Constructor

Public Member Functions	
	I2C (PinName sda, PinName scl)
	Create an I2C Master interface, connected to the specified pins. More...
	I2C (const i2c_pinmap_t &static_pinmap)
	Create an I2C Master interface, connected to the specified pins. More...

```
// Create the I2C master object
I2C I2c_m(A4, A5);  // I2C3: SDA, SCL
```

I2C Programming

- Member Functions (Methods)

void	frequency (int hz)
	Set the frequency of the I2C interface. More...
int	read (int address, char *data, int length, bool repeated=false)
	Read from an I2C slave. More...
int	read (int ack)
	Read a single byte from the I2C bus. More...
int	write (int address, const char *data, int length, bool repeated=false)
	Write to an I2C slave. More...
int	write (int data)
	Write single byte out on the I2C bus. More...
void	start (void)
	Creates a start condition on the I2C bus. More...
void	stop (void)
	Creates a stop condition on the

virtual void	lock (void)
	Acquire exclusive access to this I2C bus. More...
virtual void	unlock (void)
	Release exclusive access to this I2C bus. More...
int	transfer (int address, const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length, const event_callback_t &callback, int event=I2C_EVENT_TRANSFER_COMPLETE, bool repeated=false)
	Start nonblocking I2C transfer. More...
void	abort_transfer ()
	Abort the ongoing I2C transfer. More...

```
// Configure the I2C master object
I2c_m.frequency(BAUD);  // 100KHz baud rate

// loop through consecutive transmit values
while(1){
    I2c_m.start();
    I2c_m.write(ADDR_S | 0);   // transmit address - write
    I2c_m.write(0x33);
    I2c_m.write(0x77);
    I2c_m.stop();
```

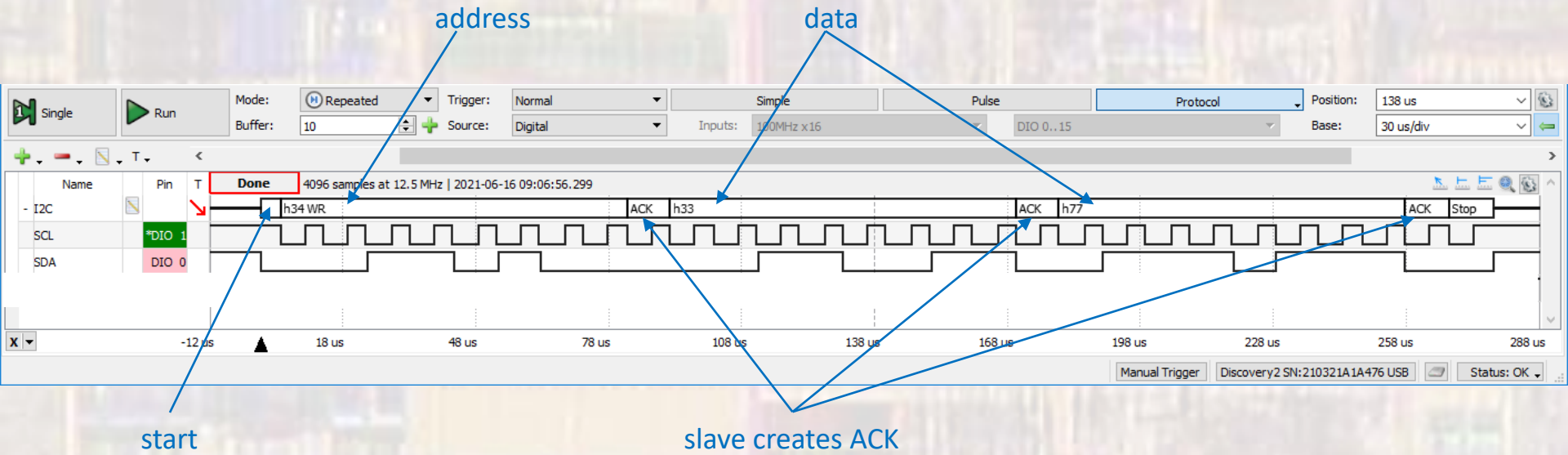
I2C Programming

- Simple example 1
  - Direct write

```
////////////////////////////////////
//
// i2c_class_ex_1 project
//
// created 6/14/21 by tj
// rev 0
//
////////////////////////////////////
//
// I2C example file for class
//
// I2C master method 1 - direct write
// using a seperate slave
// uses AD2 to see I2C writes
//
////////////////////////////////////

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

#define BAUD 100000 // 100KHz
#define ADDR_S 0x34 // corresponds to 0x1A on the MSP432 as slave

// Global HARDWARE Objects
// Create the I2C master object
I2C I2c_m(A4, A5);  // I2C3: SDA, SCL

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

    // splash
    printf("\n\ni2c_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);

    // Configure the I2C master object
    I2c_m.frequency(BAUD);  // 100KHz baud rate

    // loop through consecutive transmit values
    while(1){
        I2c_m.start();
        I2c_m.write(ADDR_S | 0);   // transmit address - write
        I2c_m.write(0x33);
        I2c_m.write(0x77);
        I2c_m.stop();

        wait_us(2000);
    } // end while

    return 0;
}// end main
```

I2C Programming

- Simple example 1
  - Direct write



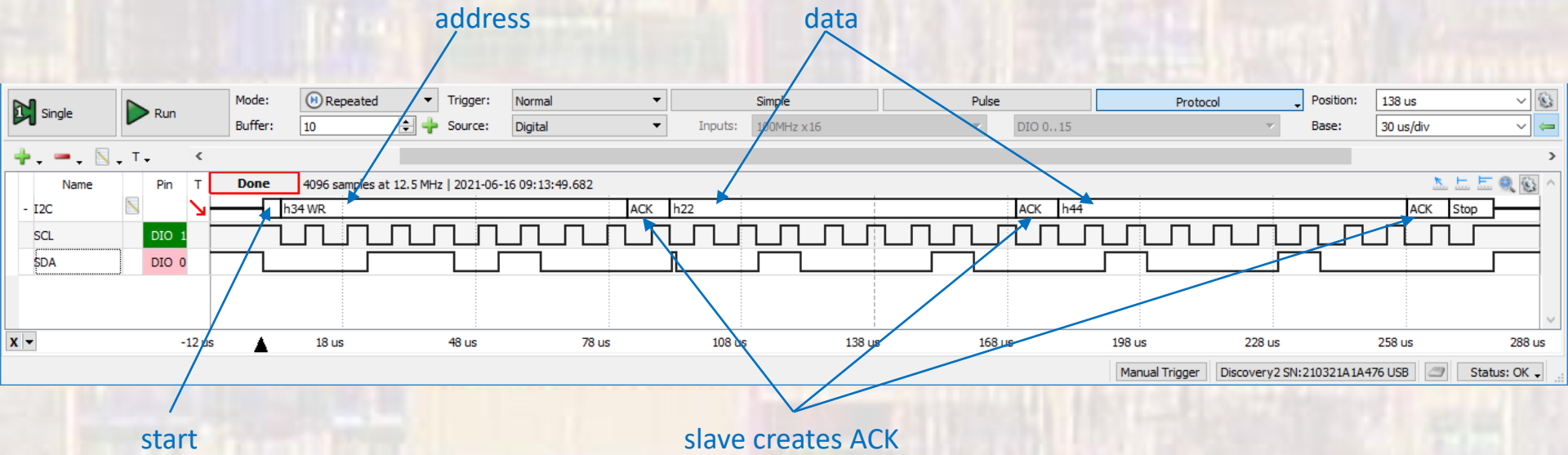
I2C Programming

- Simple example 2
  - Array write

```
////////////////////////////////////
//
// i2c_class_ex_2 project
//
// created 6/14/21 by tj
// rev 0
//
////////////////////////////////////
//
// I2C example file for class
//
// I2C master method 2 - array write
// using a seperate slave
// uses AD2 to see I2C writes
//
////////////////////////////////////

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

#define BAUD 100000 // 100KHz
#define ADDR_S 0x34 // corresponds to 0x1A on the MSP3432

// Global HARDWARE Objects
// Create the I2C master object
I2C I2c_m(A4, A5);  // I2C3: SDA, SCL

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

    // splash
    printf("\n\ni2c_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
    char foo[2] = {0x22, 0x44};

    // Configure the I2C master object
    I2c_m.frequency(BAUD);  // 100KHz baud rate

    // loop through consecutive transmit values
    while(1){
        I2c_m.start();
        I2c_m.write(ADDR_S, foo, 2);
        I2c_m.stop();

    } // end while

    return 0;
}// end main
```

I2C Programming

- Simple example 2
  - Array write



I2C Programming

- I2C Slave Class

Public Member Functions	
	I2CSlave (PinName sda, PinName scl)
	Create an I2C Slave interface, connected to the specified pins. More...
	I2CSlave (const i2c_pinmap_t &static_pinmap)
	Create an I2C Slave interface, connected to the specified pins. More...
void	frequency (int hz)
	Set the frequency of the I2C interface. More...
int	receive (void)
	Check if this I2C Slave has been addressed. More...
int	read (char *data, int length) Uses char *
	Read specified number of bytes from an I2C master. More...
int	read (void) Does not seem to work
	Read a single byte from an I2C master. More...
int	write (const char *data, int length) Uses char *
	Write to an I2C master. More...
int	write (int data)
	Write a single byte to an I2C master. More...
void	address (int address)
	Set the I2C slave address. More...
void	stop (void)
	Reset the I2C slave back into the known ready receiving state. More...

I2C Programming

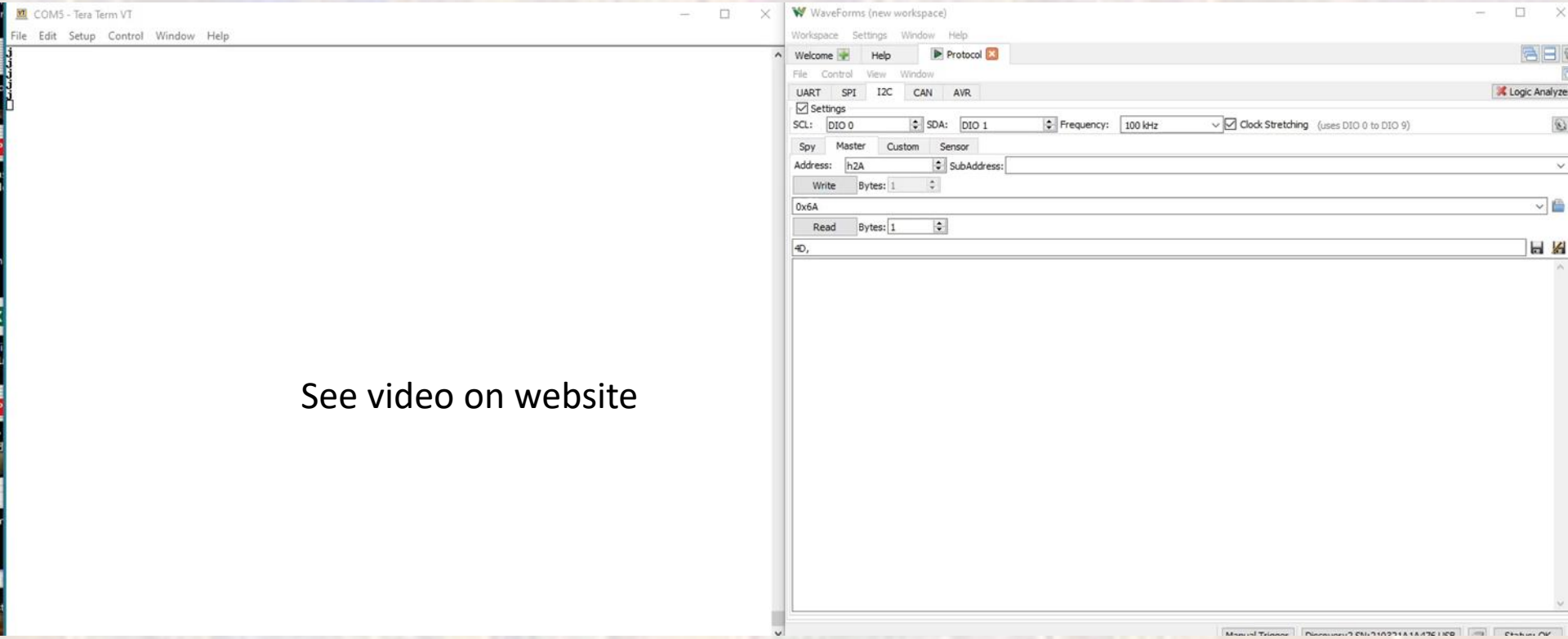
- Simple Example – 1 byte transfers

```
////////////////////////////////////
//
// i2c_class_ex_3 project
//
// created 6/14/21 by tj
// rev 0
//
////////////////////////////////////
//
// I2C example file for class
//
// I2C Slave - read
// AD2 as manager
//
////////////////////////////////////

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

#define T_WAIT 5000      // in us - 5ms
#define BAUD 100000      // 100KHz
#define ADDR_W 0x54      // Only uses upper 7 bits
                         // so use 0x2A for AD2

// Global HARDWARE Objects
// Create the I2C worker object
I2CSlave I2c_w(D14, D15);  // I2C1: SDA, SCL

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

    // splash
    printf("\n\ni2c_class_ex_3 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
            MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION)

    // local variables
    char foo;
    char boo = 'A';

    // Configure the I2C objects
    I2c_w.frequency(BAUD);  // 100KHz baud rate
    I2c_w.address(ADDR_W);
```

```
    // worker loop - checks for incoming messages
    // on write - prints the value
    // on read - provides a value then increments
    while(1){
        switch(I2c_w.receive()){
        case I2CSlave::WriteAddressed:
            I2c_w.read(&foo, 1);
            printf("%c\n", foo);
            break;
        case I2CSlave::ReadAddressed:
            I2c_w.write(&boo, 1);
            boo++;
            break;
        default:
            break;
        }// end switch
    } // end while

    return 0;
}// end main
```

I2C Programming

- Simple Example – 1 byte transfers - video
  - Using the AD2 to write and read



See video on website

I2C Programming

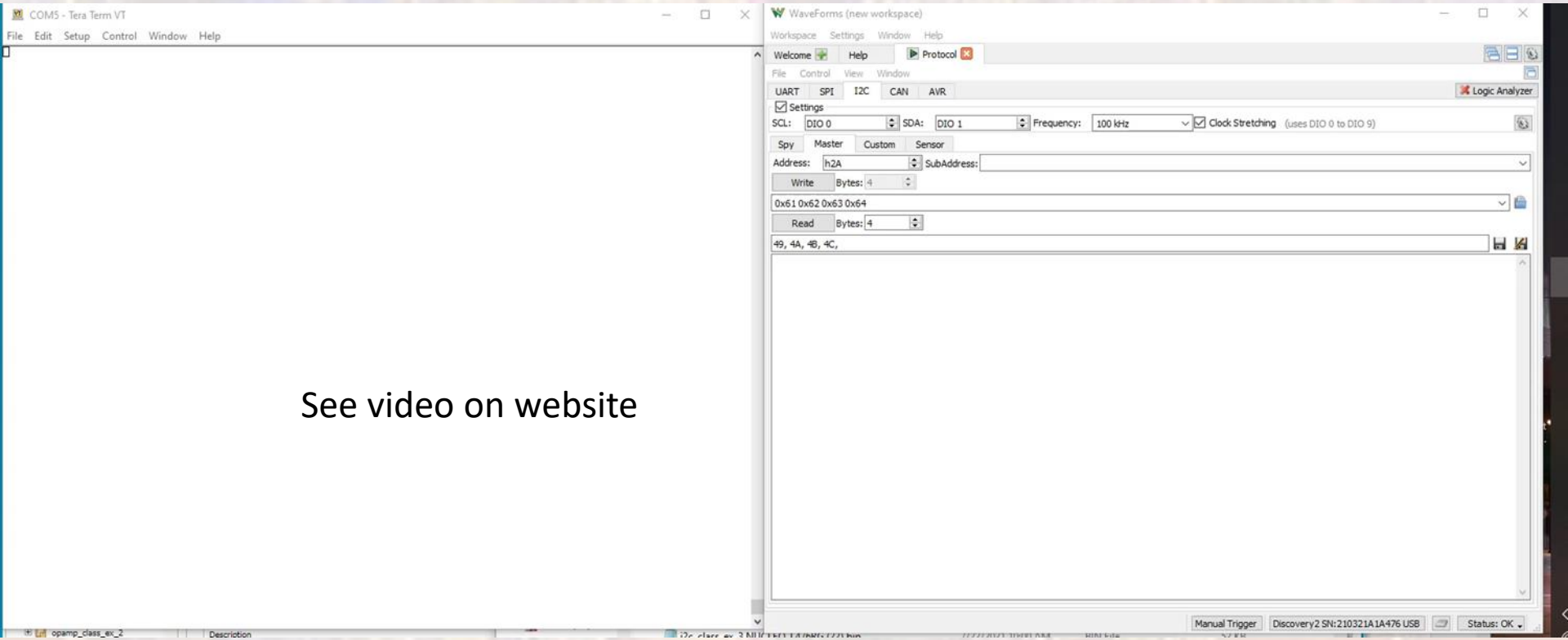
- Simple Example – 4 byte transfers

```
////////////////////////////////////
//
// i2c_class_ex_4 project
//
// created 6/14/21 by tj
// rev 0
//
////////////////////////////////////
//
// I2C example file for class
//
// I2C Slave - read
// AD2 as manager
//
////////////////////////////////////

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

#define T_WAIT 5000      // in us - 5ms
#define BAUD 100000      // 100KHz
#define ADDR_W 0x54      // Only uses upper 7 bits
                         // so use 0x2A for AD2

// Global HARDWARE Objects
// Create the I2C worker object
I2CSlave I2c_w(D14, D15);  // I2C1: SDA, SCL

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

    // splash
    printf("\n\ni2c_class_ex_4 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
            MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // local variables
    int i;
    char foo[4];
    char boo[4] = {'A', 'B', 'C', 'D'};

    // Configure the I2C objects
    I2c_w.frequency(BAUD);  // 100KHz baud rate
    I2c_w.address(ADDR_W);
```

```
    // worker loop - checks for incoming messages
    // on write - prints the value
    // on read - provides a value then increments
    while(1){
        switch(I2c_w.receive()){
        case I2CSlave::WriteAddressed:
            I2c_w.read(foo, 4);
            printf("%c %c %c %c\n", foo[0], foo[1], foo[2], foo[3]);
            break;
        case I2CSlave::ReadAddressed:
            I2c_w.write(boo, 4);
            for(i = 0; i < 4; i++)
                boo[i] = boo[i] + 4;
            break;
        default:
            break;
        }// end switch
    } // end while

    return 0;
}// end main
```

I2C Programming

- Simple Example – 4 byte transfers - video
  - Using the AD2 to write and read



I2C Programming

- Limitations summary
  - It looks like an access takes ~500us
  - Limited operating frequencies
    - 100KHz, 400KHz, 1MHz max clock rate?
  - No internal pullup capability
  - Some functions may not work

I2C Programming

- Critical Note:
  - The address used in Mbed is the full byte
    - with the rd/wr bit ignored
  - Most systems use just the 7 bits
    - The rd/wr bit is managed separately
  - Address 0x1A for most systems will be 0x34 in Mbed
    or 0x35 in Mbed

x001 1010	0011 0100
x001 1010	0011 0101