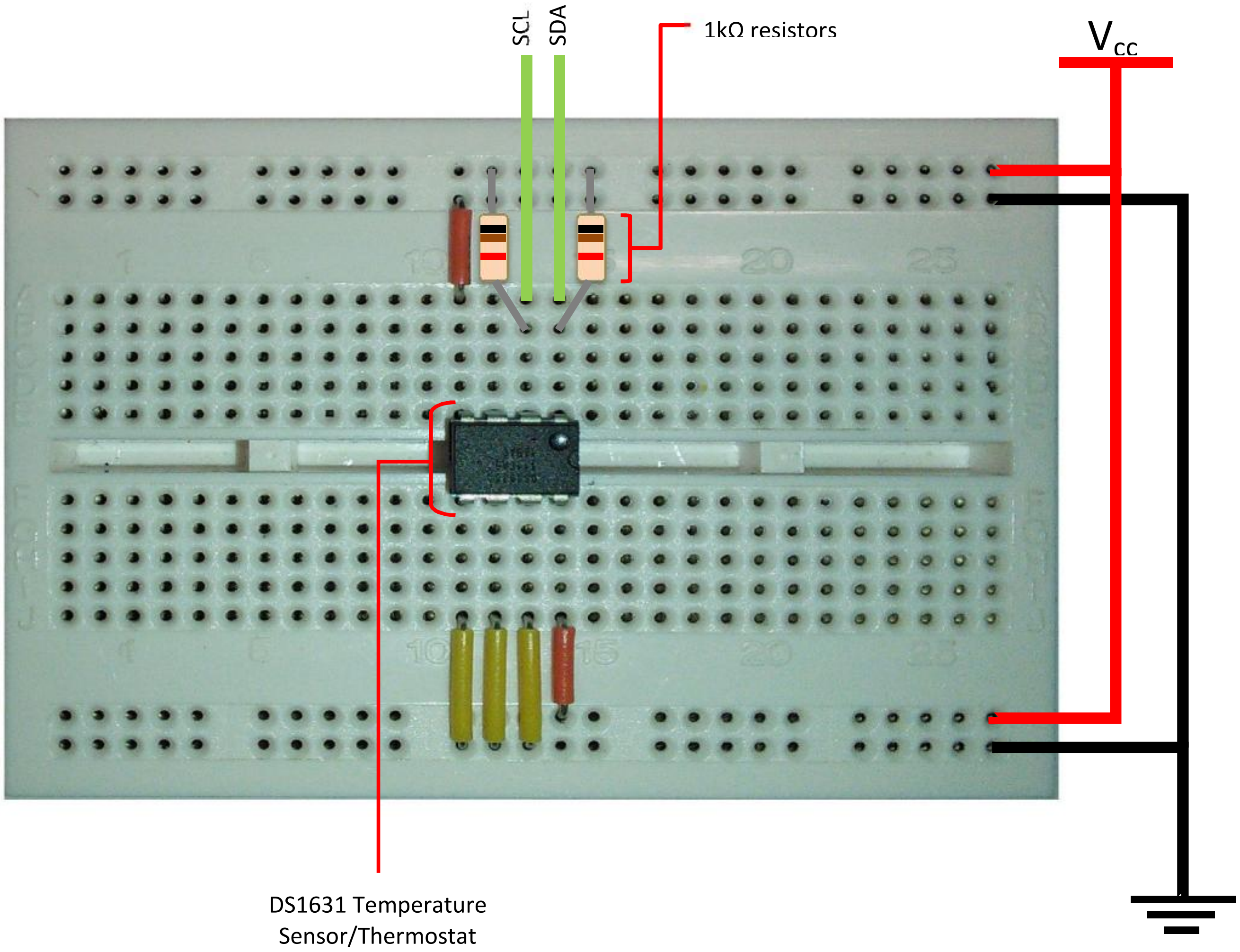
LAB EXERCISE:
SERIAL COMMUNICATION

OVERVIEW

In this lab, you will use two types of serial communication to interface with two peripherals:

- Use I2C to interface to a temperature sensor
- Use UART to interface with a PC

After each peripheral is tested separately, you need to build an integrated application, in which the PC terminal displays the current temperature. High-level functions will be used in this lab.

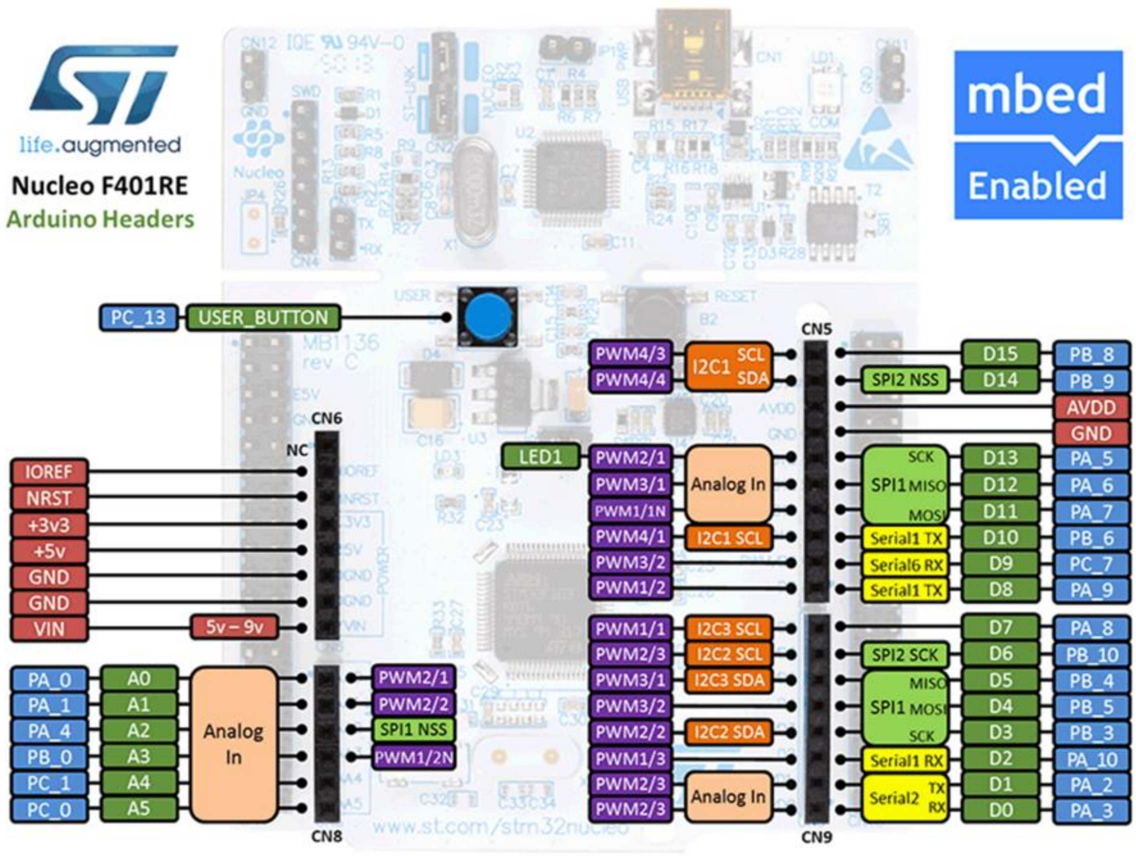


IMPLEMENTATION DETAILS

HARDWARE

NUCLEO F401RE BOARD

The Nucleo F401RE board pin descriptions are shown below:



Pin	Pin name in mbed API
Temperature sensor I2C SCL	PB_8
Temperature sensor I2C SDA	PB_9
USB UART TX	PA_2
USB UART RX	PA_3

TEMPERATURE SENSOR

The temperature (DS1631) can be accessed by I^{2}C interface.

General I^{2}C information:

- All data is transmitted MSb first over the 2-wire bus
- One bit of data is transmitted on the 2-wire bus each SCL period
- Pull-up resistors are required on SDA and SCL lines, so that when the bus is idle both lines must remain in a logic-high state

To use it, you first need to setup the address for the temperature sensor. It is done by connecting pins 5, 6 and 7 to either Vcc or ground. In this case, we'll connect pins 5, 6 and 7 to ground, which means that our temperature sensor address will be 0x90.

In this example, two 1kΩ pull-up resistors were used to keep the SDA and SCL lines in a logic-high while the bus is idle.

Each read or write command must start with a Control Byte:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	1	A_2	A_1	A_0	R/W

The R/W bit is set by the API, so you don't need to worry about it.

Command set for DS1631:

Command	Command in Hex	Description
Start Convert T	0x51	Initiates temperature conversions
Stop Convert T	0x22	Stops temperature conversions when the device is in continuous conversion mode
Read Temperature	0xAA	Reads the last converted temperature value from the 2-byte temperature register
Access TH	0xA1	Reads or writes the 2-byte TH register
Access TL	0xA2	Reads or writes the 2-byte TL register
Access Config	0xAC	Reads or writes the 1-byte configuration register
Software POR	0x54	Initiates a software power-on-reset (POR), which stops temperature conversions and resets all registers and logic to their power-up states. The software POR allows the user to simulate cycling the power without actually powering down the device

The temperature register has 16 bits, divided into MSByte and LSByte, the data is aligned from MSByte to the 3 MSBs of the LSByte, as shown below:

MSByte								LSByte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X	X

The MSB is used to indicate the sign of the temperature, for example:

- If the Temp data MSByte bit D10 = 0, then the temperature is positive and Temp value (°C) = +(Temp data) x 0.125 °C.
- If the Temp data MSByte bit D10 = 1, then the temperature is negative and Temp value (°C) = ° (two's complement of Temp data) x 0.125 °C.

The detailed information can be found at the product datasheet:

http://datasheets.maximintegrated.com/en/ds/DS1631-DS1731.pdf

SOFTWARE FUNCTIONS

Functions maybe used in this section are listed below:

Function name	Description
UART functions	
Serial (PinName tx, PinName rx, const char *name=NULL)	Create a Serial port, connected to the specified transmit and receive pins
void baud (int baudrate)	Set the baud rate of the serial port
void format (int bits=8, Parity parity=SerialBase::None, int stop_bits=1)	Set the transmission format used by the serial port
int readable ()	Determine if there is a character available to read
int writeable ()	Determine if there is space available to write a character
void attach (void(*fptr)(void), IrqType type=RxIrq)	Attach a function to call whenever a serial interrupt is generated
void send_break ()	Generate a break condition on the serial line
void set_flow_control (Flow type, PinName flow1=NC, PinName flow2=NC)	Set the flow control type on the serial port
int putc(int ch, FILE *stream)	Writes the character ch to stream. Function returns the character written, or EOF if an error happens
int getc(FILE *stream)	Read a character from the stream, an EOF indicates the end of file is reached
int printf(const char *format, ...)	Prints output both text string and data, according to format and other arguments passed to printf()
SPI functions	
SPI (PinName mosi, PinName miso, PinName sclk, PinName _unused=NC)	Create a SPI master connected to the specified pins
void format (int bits, int mode=0)	Configure the data transmission format
void frequency (int hz=1000000)	Set the spi bus clock frequency
virtual int write (int value)	Write to the SPI Slave and return the response
I2C functions	
I2C (PinName sda, PinName scl)	Create an I2C Master interface, connected to the specified pins
void frequency (int hz)	Set the frequency of the I2C interface
int read (int address, char *data, int length, bool repeated=false)	Read from an I2C slave
int read (int ack)	Read a single byte from the I2C bus
int write (int address, const char *data, int length, bool repeated=false)	Write to an I2C slave
int write (int data)	Write single byte out on the I2C bus
void start (void)	Creates a start condition on the I2C bus
void stop (void)	Creates a stop condition on the I2C bus

YOUR APPLICATION CODE

UART

Send text to the PC

- Set the baudrate
- Print “Hello mbed” to the PC
- Open a terminal (e.g. putty) on the PC to view the message

I2C

Display the temperature on the PC

- Write the Start Convert T command to the sensor, then write the Read Temperature command to the sensor
- Read the 16-bit temperature data
- Convert the temperature data into real temperature
- Print the temperature to the PC via UART