Readme

*The following is a tutorial on how to send data from a Teensy 4.0 MCU (or any other Arduino like device) to Matlab via a USB (hardware communication)*

Part 1: Arduino – Matlab communication via hardware serial (USB)

Description: three ASC11 type fields will be exported [time, incrementing sequence, voltage]; these values will be displayed in Matlab’s command window.

Matlab Script: tutorial\_arduino\_read\_write.m

Matlab version: *R2019b*

**IMPORTANT: in order for the following code to function as intended, the serial monitor of the Arduino application must be closed,** this is because the data can only be outputted to one channel.

s = serialport('COM3', 9600); s is a serialport object that represents a serial client for communication with the serial port. After creating the object, use dot notation to set its properties.

readline(s) consecutively reads lines written by Arduino, it will wait for a new one if one is not provided and the queue is empty.

writeline(s, "0.5 0") will write a line to object s (in this case, our specified Arduino communication) using strings in “ “.

A small loop was created to read 10 lines of the Arduino outputs.

Arduino Script: tutorial\_analog\_hardwareserial

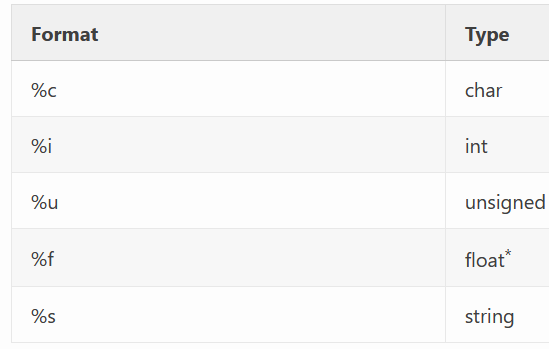
**seq++** increments variable “seq” by 1

**uint32\_t tmicros**

* The leading u says that the type of variable “tmicros” is unsigned.
* The int says that the value of variable “tmicros” is integer (not necessarily an int, though).
* The value defines the number of bits used to store the value.
* The \_t says that the sizes are standard across all platforms.
* The size of an int depends on the hardware and software being used. The size of a uint32\_t is always 32 bits. (= 4 bytes)

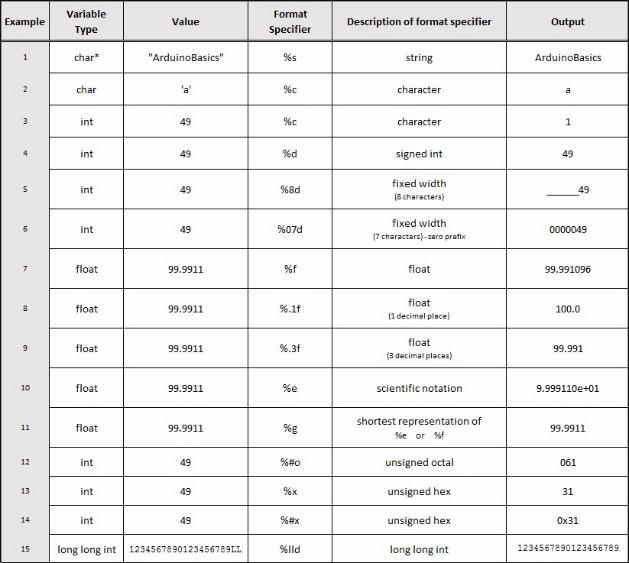
**char msg[1000]**  enables characters to variable “msg” up to 1000 characters long. (<https://www.arduino.cc/en/Reference/ASCIIchart>)

**sprintf(msg, "%lu %d %.5f\n", tmicros, seq, voltage)** follows the following format:

**sprintf** ( string $format , [mixed](https://www.php.net/manual/en/language.types.declarations.php#language.types.declarations.mixed) ...$values ) : string

where \n stands for new line feed

the rest are explained in the table below \/



Part 2: Arduino – Matlab, from matlab

Arduino Script: tutorial\_analog\_hardwareserial\_writing\_cont1

Description: Write commands on matlab that will be interpreted by the Teensy to turn on and off and LED & and control birghtness

**analogWriteResolution()** sets the resolution of the analogWrite() function. It defaults to 8 bits (values between 0-255)

**Serial.available()** Get the number of bytes (characters) available for reading from the serial port. This is data that’s already arrived and stored in the serial receive buffer (which holds 64 bytes).

\*the if statement is used as Boolean, I think any non-empty is true

**float duty = Serial.parseFloat()** returns the first valid floating point number from the Serial buffer. parseFloat() is terminated by the first character that is not a floating point number. It saves it on float variable named “duty”

**int enablepwm = Serial.parseInt();** Looks for the next valid integer in the incoming serial. Saves it on integer variable “enablepwm”

**Serial.read();** will read the data from the data byte and print a message if the data is received. The data is sent from the serial monitor to the Arduino.

**analogWrite(13,duty \* 256\* (enablepwm > 0));** this statement will allow for pin\_13 to turn on at a specified intensity (“duty”) if “enablepwm” is greater than “0”

Try this matlab inputs (one at a time) and look how pin13’s LED changes! (after establishing communication using serialport)

%% medium intensitu

writeline(s, "0.5 1")

%% low intensity

writeline(s, "0.01 1")

%% high intensity

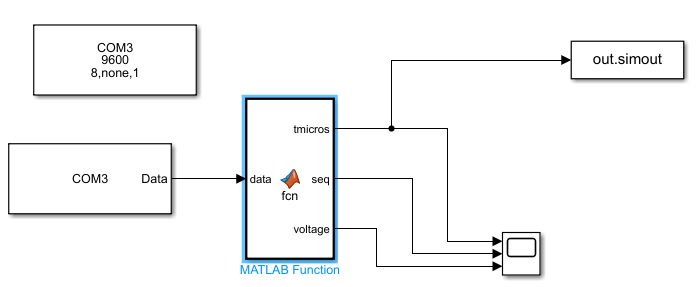
writeline(s, "0.9 1")

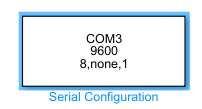
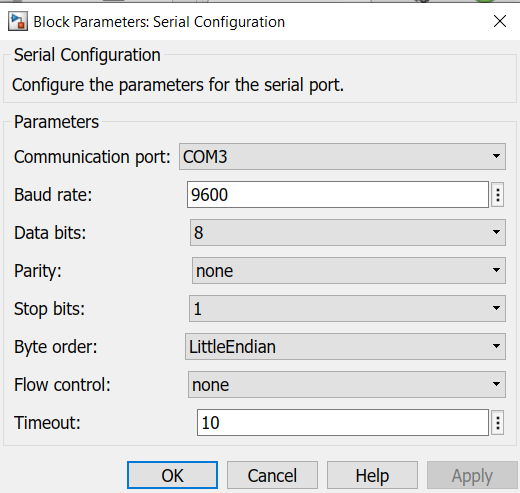
%% off

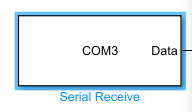
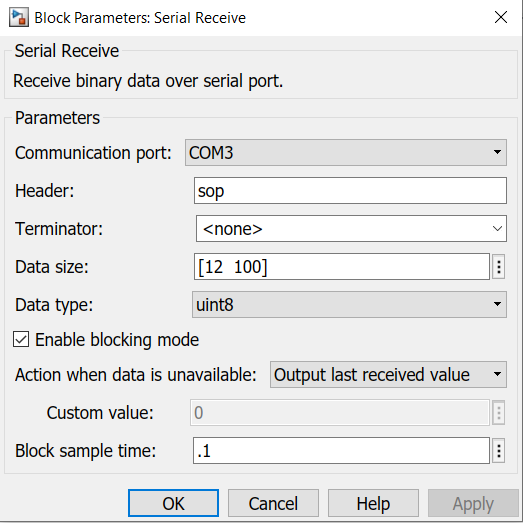
writeline(s, "0.9 0")

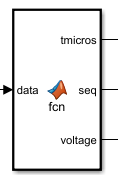
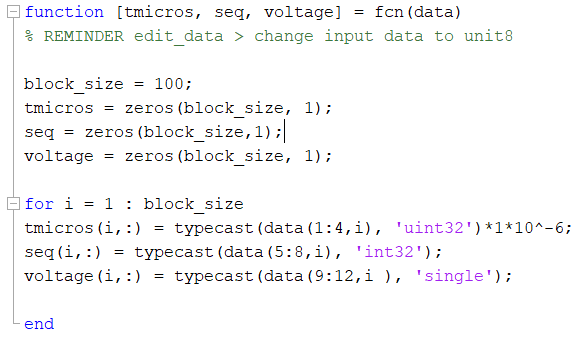
Part 3: Arduino – Simulink using byte in packets (not in ASC11)

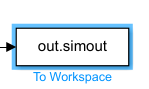
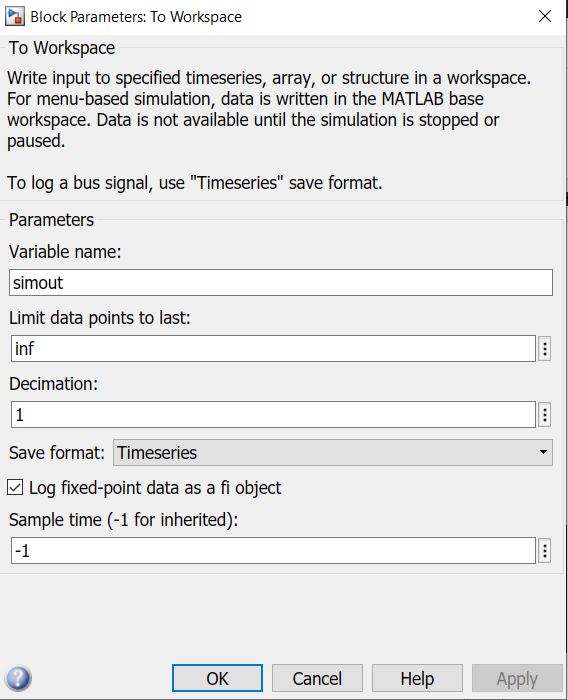
Description: Send from Teensy to Simulink, byte data in packets to achieve 1000Hz communication (without slowing down). Data will be plotted in Simulink.

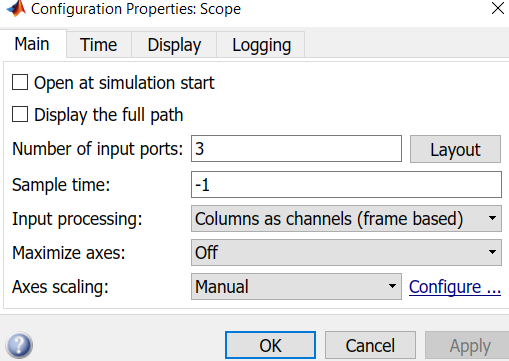
Simulink script: converting\_binary

Arduino script: SIMULINK\_hardwareserial

**struct data\_format**

**{**

**uint32\_t tmicros;**

**int32\_t seq;**

**float voltage;**

**} data;**

The section of code above creates a struct variable named “data\_format” with the field name “data”, containing tmicros, seq, & float voltage.

**static uint32\_t t0 = micros();** The static keyword is used to create variables that are visible to only one function. However unlike local variables that get created and destroyed every time a function is called, static variables persist beyond the function call, preserving their data between function calls.

**if (micros()- t0 > 1000)**

**{**

**t0 += 1000;**

**data.seq++;**

**data.tmicros = micros();**

**data.voltage = analogRead(A8) \* 3.3/ 1024;**

**static int count = 0;**

**if (count++ % 100 == 0) // % means remainder**

**Serial.print("sop"); //start of packet**

**Serial.write((char\*) &data, sizeof(data));**

**//"&" is a pointer //sizeof(data) in bytes**

**// (data\_format\***

**}**

A loop is created using time. T0 will increase in increments of 1000, this will be used to control the rate at which data is read/written.

All variables are in the field ”data”

Although we will be reading data at 1000Hz, we cannot communicate and plot at this frequency consistently, this is why data must be sent in packets.

Once 100 data samples are obtained, the data will be sent.

Since a new line won’t be created 100, we defined “sop” (start of packet) to identify where a new packet will start. This is called a header. The simulink block “serial receive” will need to know the name of the header to correctly reorganize the information.

The “serial receive” block in simulink must also know what type of data size it will receive to parse it correctly. In this case, each variable is 4 bytes (4 bytes \* 3 variables = 12 bytes), and 100 data samples.

Data using the block “to workspace will be available in the work space”

I think I am missing setting that needs to be change but I currently do not recall which one it is. I will update accordingly.