# **Embedded Systems Essentials with Arm: Get Practical with Hardware**

# Module 1

KV5: Using the Mbed API in Asynchronous Serial Communication

Let’s now turn our attention to asynchronous serial communication. Mbed distinguishes between buffered and unbuffered use of UART. In this table, we can see a sub-set of functions that the Mbed API offers to configure some of these characteristics and interact with UART – in this case, buffered.

|  |  |
| --- | --- |
| **Function Name​** | **Description ​** |
| **BufferedSerial**(PinName tx, PinName rx, int baud)​ | Create a Serial port connected to the specified transmit and receive pins, with specified baud rate​ |
| ssize\_t **write**(const void \*buffer, size\_t length) ​ | Write the contents of a buffer to a file.​ |
| ssize\_t **read** (void \*buffer, size\_t length)​ | Read the contents of a file into a buffer.​ |
| void **set\_baud**(int\_baud)​ | Set the baud rate​ |
| void **set\_format**(int bits=8, Parity parity=BufferedSerial::None, int stop\_bits=1)​ | Set the transmission format used by the serial port​ |

There is a constructor that can be used to create a serial port connected to the specified transmit and receive pins, with a specified baud rate. There are options to write the contents of a buffer to a file and to read the contents of a file into a buffer. There are also functions for setting the baud rate and the transmission format used by the serial port.

Here we see a code snippet of how a UART can be used with the Mbed API.

#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​

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(19200); //set baud rate to 19200​

//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​

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

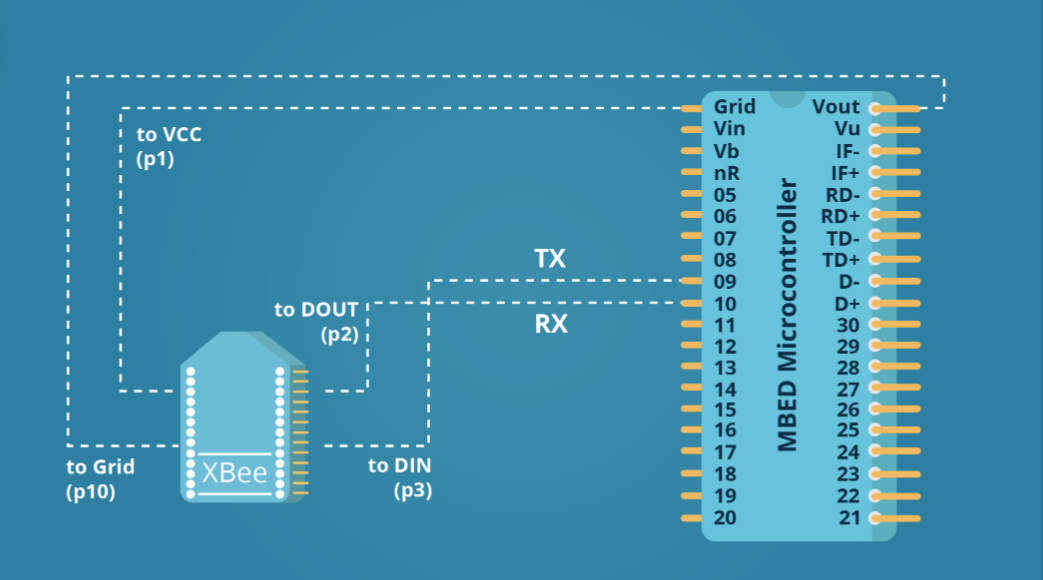
if (async\_port.readable()==1) //is there a character to be read?​

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

|  |
| --- |
| #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​  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(19200); //set baud rate to 19200​  //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​  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​  ... |

As mentioned previously the Mbed API currently differentiates buffered and unbuffered serial communication and the standard serial API has been discontinued, but for educational purposes this example uses the term “serial”. The same concept and general principle still applies, but when developing your own application the buffered or unbuffered API would need to be used.

We start by configuring a serial port, linking it to pins nine and ten for the particular target system as a UART is available on these pins. We also set up some digital in and out objects that are used to control general peripherals, as well as some character variables which will store the word we transmit and the received value. In the main function we start by setting up a baud rate of 19200 then enter a while one loop. In this loop we test a number of switches (the digital inputs) and configure a word, which we call a switch word, that we are going to send from the asynchronous port. Then, using the “put c” command, we transmit the switch word with our serial object. Finally, we determine if there is a character to be read and, if so, we read it.

This is a real world example of what we can do with asynchronous communication from an Mbed microcontroller to many other devices.

Specifically, in this example, we are communicating between an Mbed microcontroller and an XBee module using an asynchronous serial link. The XBee makes use of the Zigbee protocol, which is a wireless protocol that enables its use in low power, distributed systems, for example when creating a smart building – an internet of things application. In this diagram, pins nine and ten are linked to the XBee as tx and rx, as well as some links for power from the microcontroller. We now have the capability to send and receive wireless communication from the connected system.

**Visit arm.com for more information.**