MIL: Introduction to UART

By Marquez Jones

Communications in Embedded Systems:

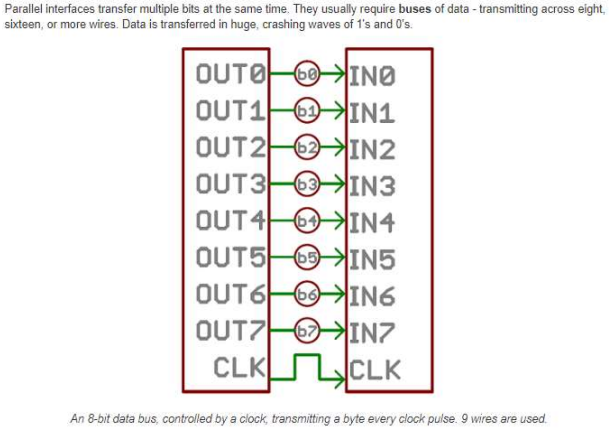
PROBLEM STATEMENT:

In broad field of embedded systems, we commonly find a need to transmit data to and from whatever electronic design we have. This holds true in robotics and in MIL where most of our PCBs, either need to report data back to the motherboard or will continually require to receive commands from other parts of the system.

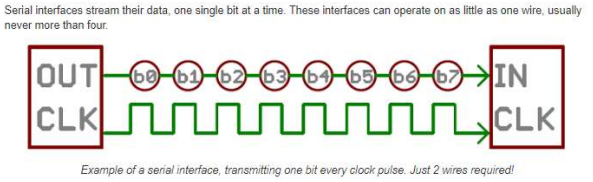
The most common form of communications in MIL is CAN (Control Area Network),but for the purposes of learning, I'll be going over UART in this document which is also a very important communication protocol to learn. Both CAN and UART are serial communication protocols.

SERIAL VS PARALLEL:

FREE LEARNING(pic related): https://learn.sparkfun.com/tutorials/serial-communication/all



An 8-bit data bus, controlled by a clock, transmitting a byte every clock pulse. 9 wires are used.



Example of a serial interface, transmitting one bit every clock pulse. Just 2 wires required!

Looking at the diagram above, parallel communications sends all the data at once. Multiple bits of data are sent at once to the receiving processor. In the serial case, we send a bit one at time. As a rule, parallel is much faster than serial and is how our processor talks internally to memory systems and other peripherals. But for external communication it creates more challenges in hardware design as it now means we need more physical connections to transmit data. This is why we use serial. Instead of having to make many connections across PCBs, we can just use 1 or 2 wires instead.

Polled UART Example:

DESC:

In this example, I will only go over the UART specific parts of the code as opposed to the entire example. This code will introduce you to deploying UART on our designs along with showing you some of the available function in the MIL_TIVA library developed to expediate writing software on our MCU.

UART INIT:

```
/***************UART INIT START************************/

//initialize UART
MIL_InitUART(UART1_BASE, MIL_DEFAULT_BAUD_115K);

/***************UART INIT END*************************/
```

This is the function call for UART init. Here we tell the function which UART module we're using which can be found in the manual. There are 8 total UART modules we can use UART0 to UART7. Each of these will run independently from each other. Here I only use UART 1. The second parameter is the communication speed(baud rate) which tells us how many bits per second we will send. Here I set it to 115 kbps. In Asynchronous communication protocols like UART, anything talking to our design here must also be talking at 115kbps.

UART POLLING:

```
/***************DATA START*****************************/

 //C string example
 const char cstring[] = "By Marquez Jones";

/***************DATA END*******************************/

 MIL_UART_OutCString(UART1_BASE,cstring);

 while(1){

     /************POLLED VERSION OF THE CODE***************/
    if(UARTCharsAvail(UART1_BASE)){

        //read received data
        uint8_t rx_data = UARTCharGet(UART1_BASE);

         //transmit data
        UARTCharPut(UART1_BASE,rx_data);

    }
}
```

Here we have demonstrations of both receiving and transmitting data. In order to show that the UART works I first send my name via the MIL_UART_OutCString function which will transmit the string I have declared above which is my name. I recommend searching C strings if you're not familiar. Unlike other programming languages, C does not have a string object. Strings are implemented as character arrays terminated by a null character. When I declared it above, the double quotes told the compiler to make the array one longer than "needed" so it had space to place the null.

The next portion of the code is our while loop. In this, it uses UARTCharsAvail provided by the TI TivaWare library to constantly check if new data has been received. If it is received, I send it back in order to echo it. The receive is done by the UARTCharGet which returns an 8 bit number and the transmit is done by UARRCharPut which is send an 8 bit number.