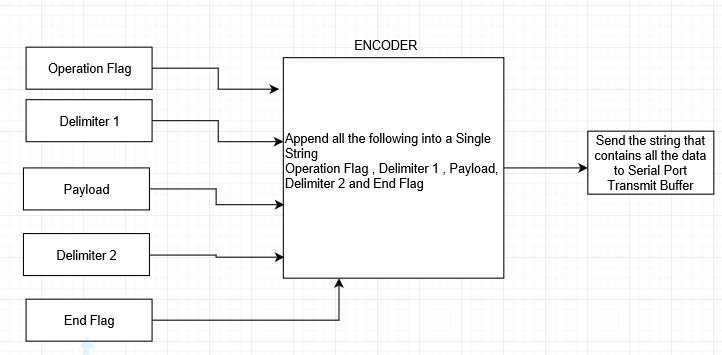
GUI ENCODER:



The C# Gui allows user to send various commands to Microcontroller. The C# GUI code composer class to display what data is being sent from Microcontroller to Gui in character and hex format. The Gui also allows user to manually type commands and send commands like a terminal. When user presses any buttons in the C# GUI, the respective command is encoded into a data packet string which contains operation flags, delimiters, payload and end flag. Finally, that string is sent to Microcontroller.

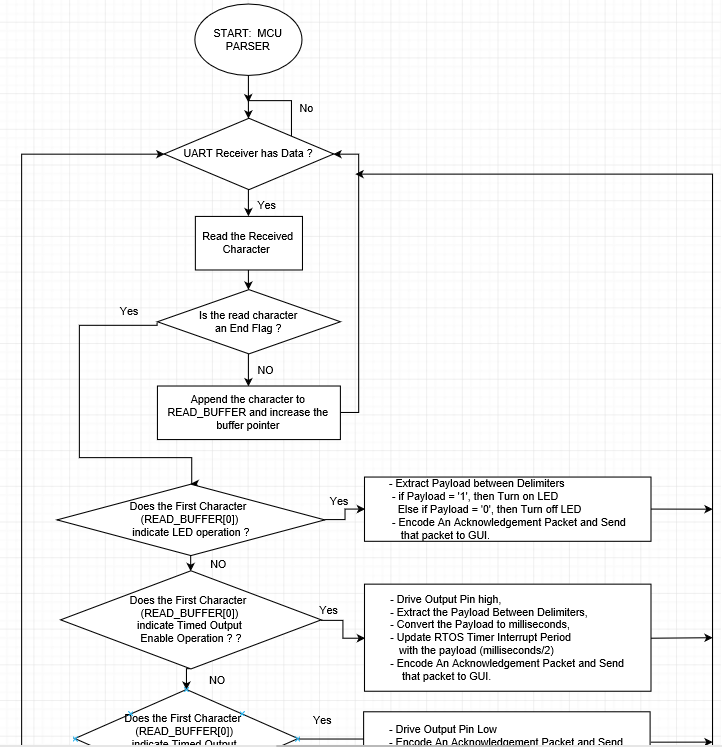
A example of encoding process would look like, (See Data Packet Design For all available commands)

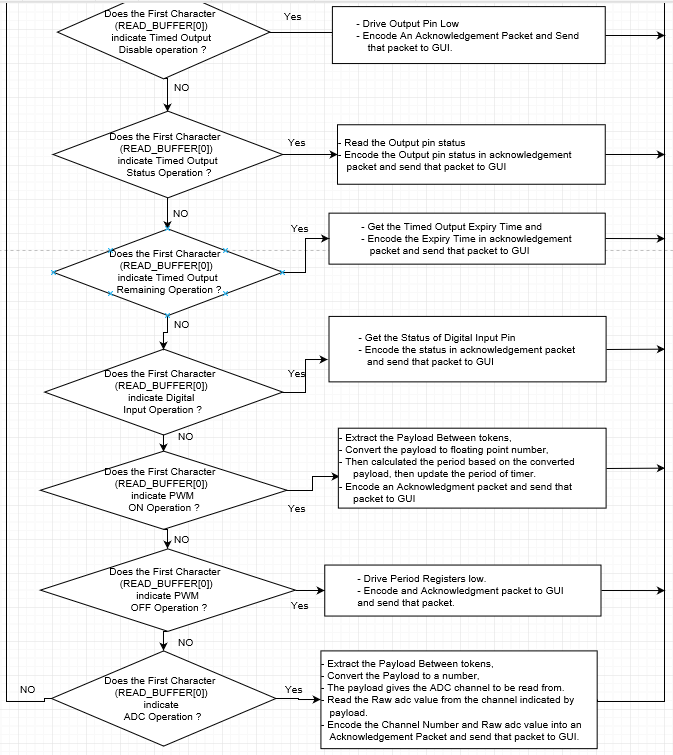
User Clicks Led On Operation, the encoder concatenates

‘L’ (Operation Flag ) , ‘^’ (Delimiter 1), ‘1’ (Payload) , ‘&’ (Delimiter 2), ‘@’ (End Flag ) into a single string

Which looks like “L^1&@”. This string then gets send to serial port Transmit Buffer.

MCU PARSER:





The application first initializes all the tasks and then keeps looping on the respective task that been scheduled to run. Majority of the application is focused on one task which handles all the commands received.

As shown by flowchart, the application stored stores all the commands into a buffer. When the end flag is received, based on the operation flag (part of command packet) respective operation are performed.

In general, when an end flag is received, then depending upon operation flag, the payload is extracted from data packet and necessary operation are performed. Once the operations are performed then the result of those operations are stored in a response packet and then sent to GUI.

A simple LED ON operation would look like,

When ‘L’ is received , it is stored in READ\_BUFFER[0],

When ‘^’ is received, it is stored in READ\_BUFFER[1],

When ‘1’ is received, it is stored in READ\_BUFFER[2],

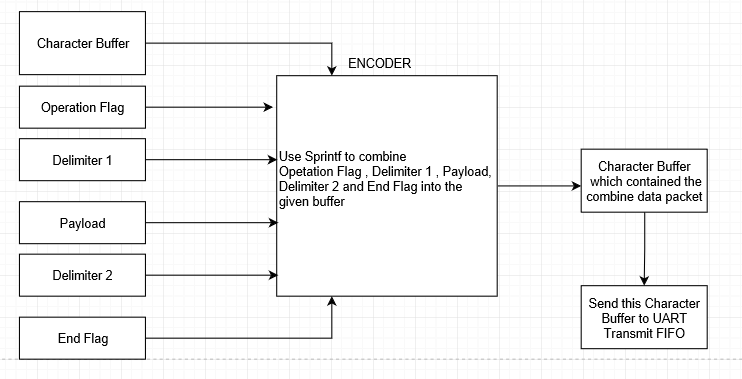
When ‘&’ is received, it is stored in READ\_BUFFER[3],

Finally, when End Flag ‘@’ is received

The Code checks the What is Store in READ\_BUFFER [0], which usually contains Operation flag,

For this case Operation Flag is ‘L’, which is Led. Then once this is detected, the code extracts the payload between delimiters (^, &) from the buffer, which is this case is ‘1’, When ‘1’ is detected the code turns on the led. Once the Operation is complete, the code sends an acknowledge data pack. Which is discussed in MCU Encoder.

MCU Encoder:



Once an operation is complete, the MCU creates a response packet and then transmit to GUI.

For example, Led OnOperation is requested by, Once Operation is complete, the encoder creates a Character buffer using sprintf function provided by <string.h>

‘L’ (Operation Flag),  
’^’ (Delimiter 1),  
’1’ (Payload),  
’&’ (Delimiter 2),  
‘#’ (End Flag)  
these characters are stored into a character buffer as “L^1&#” and the send to GUI.

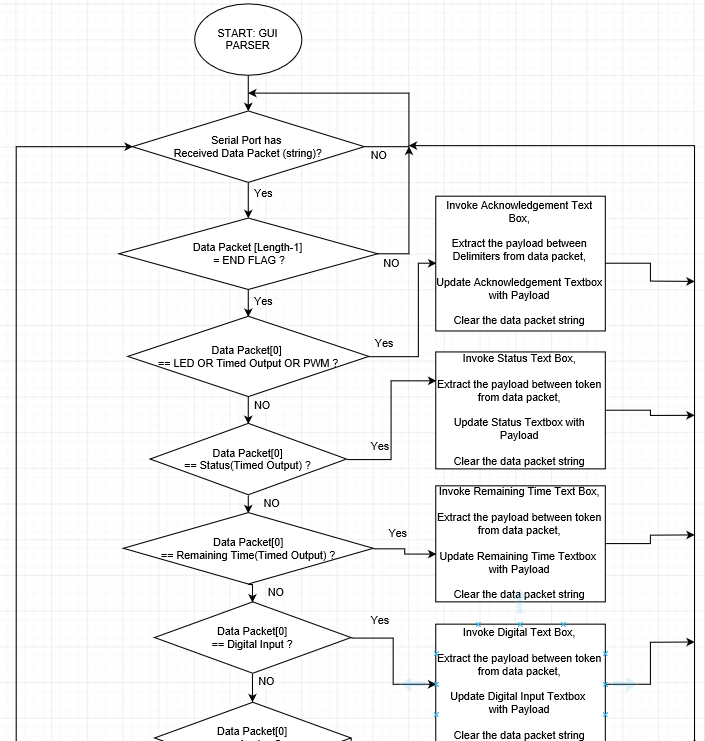
For ADC operation the encoded packet looks likes

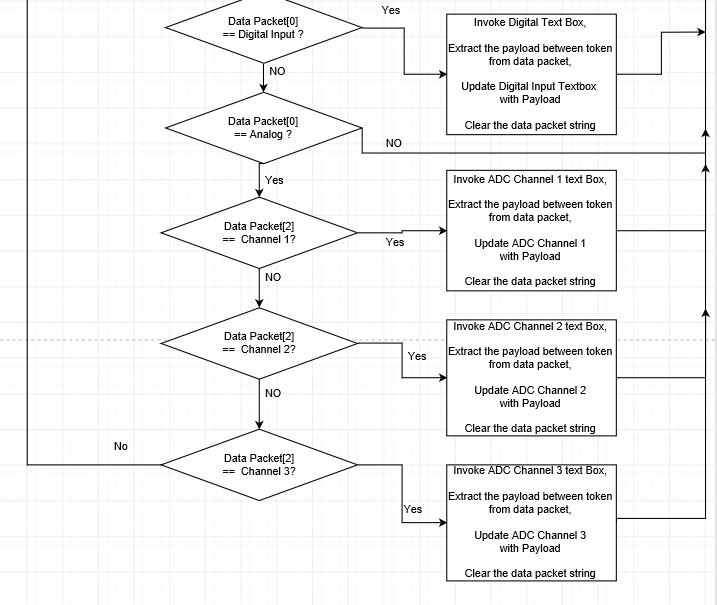
‘A’ (Operation Flag)  
‘^’ (Delimiter 1)  
‘1’ (Payload 1), inidicating channel 1 and textbox channel 1 for GUI

‘&’ (Delimiter 2),   
‘(Raw ADC Value for example . 1023)’,

‘\*’ (Delimiter 3),  
‘#’ (End Flag)  
these characters are stored into a character buffer as “A^1&1023\*#” and then sent to GUI.

GUI PARSER:





The C# GUI has its own Serial Port Data receive handler which gets triggered when data is received. As the flowchart shows all the data received is stored into a data packet buffer (string). Then code scans the string to see whether an end flag has been received. If an end flag has been received, then code checks for operation flag on the same string and depending upon operation flag on the string the code invoke separate threads to update various GUI elements like textboxes and others.

For Example, ADC operation would look like

Assuming “A^1&1023\*#” has been received in string

The code scans the string for Operation flag in 0th index of string which is ‘A’ then code checks to see channel number, 3rd index of string which is ‘1’, then code invokes ADC channel 1 textbox in a separate thread which scan for payload between delimiter2 and 3, in this case it extracts what is between ‘&’ and ‘\*’. Then the payload between delimiters, in this case 1023, is display in the textbox.

Dependencies

The following chart shows which headers files and implemented are used for source file used for MCU.

