Assignment 3 Review
Task 1: Analog Input

- Many solutions:
  - Some much more/less efficient than others
  - Good:
    - Bit-Shifting
    - Unsigned Chars
    - At-most 1 divide (can be done with no divides for 8-state)
  - Bad:
    - Floating-point arithmetic
    - Unnecessary 16-bit arithmetic
  - Ugly:
    - LOOPING!!!
Task 1: Analog Input

• Example Algorithm:
  – PORTB = 0xFF >> (7-(ADR1>>5));

• Common Gotchas:
  – Don’t attempt to use ADRx for storage (Read-only)
  – ADC configuration – if polling, do it properly
  – 100 µSec powerup delay
Task 1e: Home-Brew Voltmeter

• Good:
  – Req’d resolution = 0.1v → Use an integer variable which represents 100mV per count...
  – Interrupt driven 7-seg refresh

• Bad:
  – Implicit floating point type conversion

• Ugly:
  – All maths done in floating point, use calculations for refresh-rate on 7-segs
Task 2: Interrupt driven serial

• Good:
  – Interrupt driven TX and RX.
  – Interrupts enabling and disabling as appropriate.

• Bad:
  – Polled TX (or RX) in Main.

• Ugly:
  – Polling INSIDE the ISR.
Task 2: Interrupt Driven Serial

- Many different ways to solve this problem:
  - Fundamental unit is the Buffer (Array).
    - Linear Buffer / Stack
    - Circular Buffer
- Data to be TX’d gets placed in buffer
- ISR does the following:
  - If Serial TX interrupt (TDRE == 1?):
    - Read char from Buffer++
    - If char != ‘end of buffer’ then write to SCDR
    - Else, clear TIE in SCCR1
Task 2: Pointer management...

• All good interrupt-driven serial requires a **SOLID** understanding of pointers.
  – Pointer-to-a-pointer = Arrays of pointers
  – *A pointer is just a 16-bit number*

• Copying large strings takes time, passing pointers is a lot faster.
Describing the Problem
Program Design and Buffers

```c
#include <IO6811.H> /* internal */
#include <int6811.H> /* ISR */

#define BUFFSIZE 256
char inCharBuffer[BUFFSIZE]; /*
char numCharsInBuffer = 0; /* Why?
char * outMessage = "X data received
char * charToSend; /* The char

const char END_TRANSMIT = '$';
```
• A buffer for read data to be transmitted: inCharBuffer
• Current size of Buffer: numCharsInBuffer
• Maximum Size of Buffer: BUFFSIZE
• Buffer for the Message “x data received \r\n$”: * outMessage
• Current character to be sent: * charToSend

• End of Frame: END_TRANSMIT= ‘$’;

```c
#define BUFFSIZE 256
char inCharBuffer[BUFFSIZE]; /*
char numCharsInBuffer = 0; /*
char * outMessage = "X data rec
char * charToSend; /* The char
const char END_TRANSMIT = '$';
```
ISR

• For each interrupt
  – **IF (Receive Interrupt)**
    • Store another Character in Buffer
    • ( IF Buffer is full Rewrite Last ! )
  – **IF (Transmit Interrupt)**
    • IF (END OF FRAME)
      – IF (The Buffer is not Empty)
        » Build next FRAME
      – ELSE
        » Return ( no more data to transmit)
    • Transmit current character in FRAME
    • return
const char TRANSMIT_REGISTER_EMPTY_MASK = 0x80; /* The TDRE bit. Set this bit to send a character. */
const char RECEIVE_REGISTER_FULL_MASK = 0x20;  /* The RDRF bit. Set this bit when a character is received. */
const char BAUD_9600 = 0x30;  /* 9600 baud is the fastest rate that hardware allows. */
const char SCCR1_VAL = 0x00;  /* We don't need to change this register. */
const char SCCR2_VAL = 0xAC;  /* Bit 7 is Transmit Interrupt Enable, */
   /* bit 5 is Receive Interrupt Enable, bit 3 is Transmit Enable, */
   /* bit 2 is Receive Enable. See p5-10 of manual. */

/* Load initial values in the serial configuration registers. Set up */
void InitialiseSerial()
{
    SCCR1 = SCCR1_VAL;
    SCCR2 = SCCR2_VAL;
    BAUD = BAUD_9600;

    *charToSend = END_TRANSMIT;  /* Nothing to send initially. */
}
void StartFromMessageBeginning()
{
    int i; /* Loop iterator */

    /* Grab the first character from the inCharBuffer. */
    *outMessage = inCharBuffer[0];

    /* Delete the first character in the in-message-buffer (as we will */
    for(i = 0; (i+1) < numCharsInBuffer; i++) /* Use (i+1) since the */
    {
        inCharBuffer[i] = inCharBuffer[i+1];
    }
    numCharsInBuffer--;

    /* Set the output pointer to point to the start of the string. */
    charToSend = outMessage;
}
ISR: Due to Data received

```c
interrupt void SCI_interrupt()
{
    /* There are two kinds of interrupts we need to handle - receive interrupts.
    * We will have a big IF structure that determines which interrupt has
    */
    #ifdef RECEIVE_INTERRUPT
    if((SCSR & RECEIVE_REGISTER_FULL_MASK))
    {
        inCharBuffer[numCharsInBuffer] = SCDR; /*Store the newly received character
        numCharsInBuffer++;
        if(numCharsInBuffer == BUFFSIZE)
        {
            /* If the buffer overflows, just dump everything in the last byte
            */
            numCharsInBuffer--;
        }
        PORTB = 1;
        return;
    }
    ```
ISR: In case TR buffer empty

```c
/****************************TRANSMIT INTERRUPT**************************/
if(SCSR & TRANSMIT_REGISTER_EMPTY_MASK) /* Reading the SCSR and writing to
{                                           
    if(*charToSend == END_TRANSMIT)         
        /*Just finished the previous message, or no message has been sent
        if(numCharsInBuffer > 0)          
            /*There is data to send. Start at the beginning of the message.
            StartFromMessageBeginning();    
            PORTB = 2;                   
        }else{                             
            /*No message being sent, and no message to send.*/
            PORTB = 3;                    
            return;                     
        }                                 
    }                                     
    /*Send the next character.*/
    SCDR = *charToSend;                 
    charToSend++;                       
    charToSend++;                       
    PORTB = 4;                          
    return;                             
```
int main(void)
{
    disable_interrupt();            /* equivalent to SEI assembler */
    InitialiseSerial();              /* equivalent to CLI assembler */
    enable_interrupt();             /*Busy-loop*/
    while(1);                       /*Busy-loop*/