Assignment 1 - UART

EGC 443 Embedded Systems

George Dagis (CE), NAME OMITTED (CE)

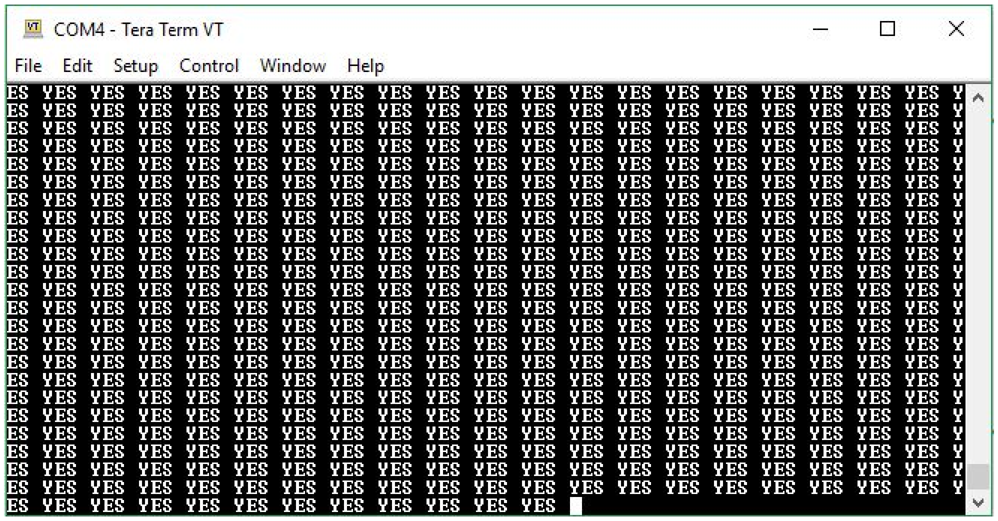
January 6th, 2018

Instructor: Michael Otis

UART0 - Transmit Code

```
/* p4_1.c: Sending "YES" to UART0 on TI ARM Launchpad (TM4C123GH6PM) */

/* UART0 is on USB/Debug */

/* Use TeraTerm to see the message "YES" on a PC */

#include <stdint.h>

#include "tm4c123gh6pm.h"

void UART0Tx(char c);

void delayMs(int n);

void SystemInit(void); //added


int main(void)

{

    SYSCTL->RCGCUART |= 1;  /* provide clock to UART0 */

    SYSCTL->RCGCGPIO |= 1;  /* enable clock to PORTA */


    /* UART0 initialization */

    UART0->CTL = 0;         /* disable UART0 */

    UART0->IBRD = 104;      /* 16MHz/16=1MHz, 1MHz/104=9600 baud rate */

    UART0->FBRD = 11;       /* fraction part, see Example 4-4 */

    UART0->CC = 0;          /* use system clock */

    UART0->LCRH = 0x60;     /* 8-bit, no parity, 1-stop bit, no FIFO */

    UART0->CTL = 0x301;     /* enable UART0, TXE, RXE */


    /* UART0 TX0 and RX0 use PA0 and PA1. Set them up. */

    GPIOA->DEN = 0x03;      /* Make PA0 and PA1 as digital */

    GPIOA->AFSEL = 0x03;    /* Use PA0,PA1 alternate function */

    GPIOA->PCTL = 0x11;     /* configure PA0 and PA1 for UART */


    delayMs(1);             /* wait for output line to stabilize */

    for(;;)

    {

        UART0Tx('Y');
        UART0Tx('E');
        UART0Tx('S');
        UART0Tx(' ');
    }
}


/* UART0 Transmit */
/* This function waits until the transmit buffer is available then */
/* writes the character in the transmit buffer. It does not wait */
/* for transmission to complete. */
void UART0Tx(char c)
{
    while((UART0->FR & 0x20) != 0); /* wait until Tx buffer not full */
    UART0->DR = c;                  /* before giving it another byte */
}
/* Append delay functions and SystemInit() here */
void delayMs(int n)
{
                int i, j;
                                for(i=0; i<n; i++)                      //decrement
                                                for (j=0; j<3180; j++)  //decrement more
                                                {}                      //do nothing
}
void SystemInit(void)
{               SCB->CPACR |= 0x00F00000;                       //Coprocessor Access Control Register
}
```

The resulting output of this code on a dummy terminal using Tera Term can be seen in Figure 1, below.



```
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES YES Y
ES YES YES YES YES YES YES YES YES YES YES YES YES
```

Figure 1: Screenshot of TeraTerm dummy terminal outputting resulting message from code

UART0 - Receive Code

```
/* p4_2.c: Read data from UART0 and display it at the tri-color LEDs. */

/* The LEDs are connected to Port F 3-1. */

/* Press any A-z, a-z, 0-9 key at the terminal emulator */

/* and see ASCII value in binary is displayed on LEDs of PORTF. */

#include <stdint.h>

#include "tm4c123gh6pm.h"

char UART0Rx(void);

void delayMs(int n);


int main(void)

{

    char c;


    SYSCTL->RCGCUART |= 1;  /* provide clock to UART0 */

    SYSCTL->RCGCGPIO |= 1;  /* enable clock to PORTA */
    SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */

    /* UART0 initialization */
    UART0->CTL = 0;         /* disable UART0 */
    UART0->IBRD = 104;      /* 16MHz/16=1MHz, 1MHz/104=9600 baud rate */
    UART0->FBRD = 11;       /* fraction part, see Example 4-4 */
    UART0->CC = 0;          /* use system clock */
    UART0->LCRH = 0x60;     /* 8-bit, no parity, 1-stop bit, no FIFO */
    UART0->CTL = 0x301;     /* enable UART0, TXE, RXE */

    /* UART0 TX0 and RX0 use PA0 and PA1. Set them up. */
    GPIOA->DEN = 0x03;      /* Make PA0 and PA1 as digital */
    GPIOA->AFSEL = 0x03;    /* Use PA0,PA1 alternate function */
    GPIOA->PCTL = 0x11;     /* configure PA0 and PA1 for UART */
    GPIOF->DIR = 0x0E;      /* configure Port F to control the LEDs */
    GPIOF->DEN = 0x0E;
    GPIOF->DATA = 0;
    for(;;)
    {
        c = UART0Rx();          /* get a character from UART */
        GPIOF->DATA = c << 1;   /* shift left and write it to LEDs */
    }
}

/* UART0 Receive */
/* This function waits until a character is received then returns it. */
char UART0Rx(void)
{
    char c;
    while((UART0->FR & 0x10) != 0); /* wait until the buffer is not empty */
    c = UART0->DR;                  /* read the received data */
    return c;                       /* and return it */
}
```

```
/* Append delay functions and SystemInit() here */
void delayMs(int n)
{
	int i, j;
		for(i=0; i<n; i++)
			for (j=0; j<3180; j++)
			{}
}
void SystemInit(void)
{
	SCB->CPACR |= 0x00F00000;
}
```

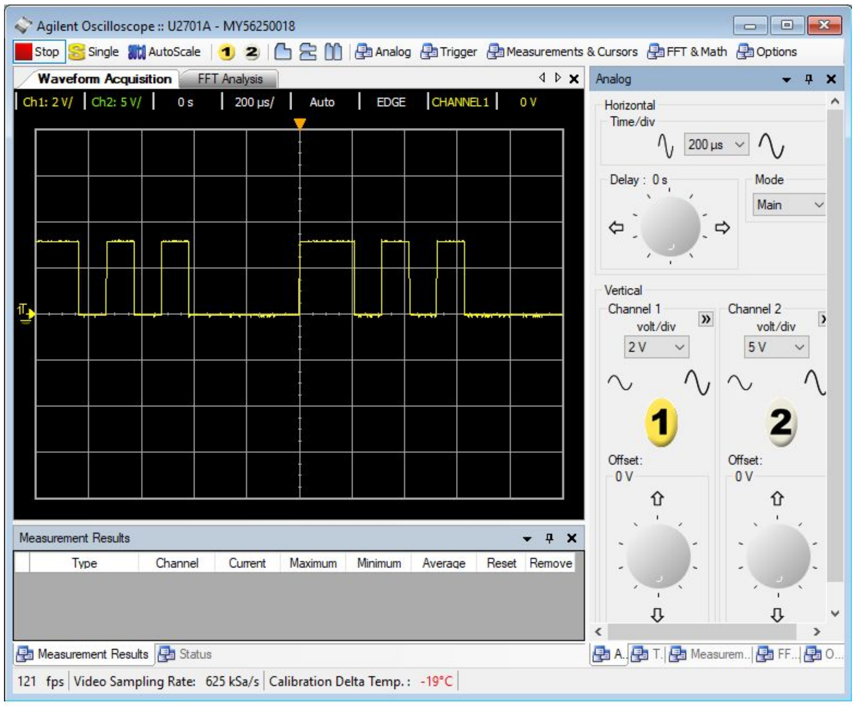
UART5 - Transmit & Receive Code

```
/* p4_3.c: Sending "Hello" to UART5 on TI ARM LaunchPad (TM4C123GH6PM) */
/* UART5 Tx is on PE5 */
/* Use TeraTerm to see the message "Hello" on a PC */

#include <stdint.h>
#include "tm4c123gh6pm.h"

void UART5Tx(char c);
void delayMs(int n);
void SystemInit(void);

int main(void)
{
	char message[] = "a";
	int i;
```

```
        SYSCTL->RCGCUART |= 0x20;  /* provide clock to UART5 */
        SYSCTL->RCGCGPIO |= 0x10;  /* Enable clock to PORTE */

        /* UART5 initialization */
        UART5->CTL = 0;                 /* disable UART5 */
        UART5->IBRD = 104;              /* 16MHz/16=1MHz, 1MHz/104=9600 baud rate */
        UART5->FBRD = 11;               /* fraction part, see Example 4-4 */
        UART5->CC = 0;       /* use system clock */
        UART5->LCRH = 0x60;             /* 8-bit, no parity, 1-stop bit */
        UART5->CTL = 0x301;             /* enable UART5, TXE, RXE */

        /* UART5 TX5 and RX5 use PE5 and PE4. Set them up. */
        GPIOE->DEN = 0x30;              /* make PE5, PE4 as digital */
        GPIOE->AMSEL = 0;               /* turn off analog function */
        GPIOE->AFSEL = 0x30;            /* use PE5, PE4 alternate function */
        GPIOE->PCTL = 0x00110000;  /* configure PE5, PE4 for UART5 */

        delayMs(1);          /* wait for output line to stabilize */

        for(;;)
        {
        for (i = 0; i < 1; i++)
        UART5Tx(message[i]);
        }
}
/* UART5 Transmit */
void UART5Tx(char c)
{
        while((UART5->FR & 0x20) != 0); /* wait until Tx buffer not full */
        UART5->DR = c;                  /* before giving it another byte */
}
/* Append delay functions and SystemInit() here */
```

```
void delayMs(int n)

{

        int i, j;

        for(i=0; i<n; i++)

                for (j=0; j<3180; j++)

                        {}

}

void SystemInit(void)

{

  SCB->CPACR |= 0x00F00000;

}
```

The resulting output waveform created from the above code can be seen in Figure 2, below.



Ch1: 2 V/ | Ch2: 5 V/ | 0 s | 200 µs/ | Auto | EDGE | CHANNEL1 | 0 V

Analog
Horizontal
Time/div 200 µs
Delay : 0 s
Mode Main
Vertical
Channel 1 volt/div 2 V Offset: 0 V
Channel 2 volt/div 5 V Offset: 0 V

Type	Channel	Current	Maximum	Minimum	Average	Reset	Remove

121 fps | Video Sampling Rate: 625 kSa/s | Calibration Delta Temp.: -19°C

Figure 2: Oscilloscope output of Program 3 - UART5

Summary

For part 1 of this experiment, my partner and I had to transmit code via UART0. For this we were required to first instantiate SystemInit(void) in the top of the main code. At the bottom of our main code we added a small SystemInit subroutine. This gives the coprocessor access to the control register. We also had to add a delay function. This basically just increments a counter and would output our transmit code at the specified delay time.

For part 2, the group had become familiar with another aspect of UART functionality - the receiving end. Similar to the code in program 1, a simple delay function was also added. In program 2 however it was not necessary to include SystemInit above the main method.

Finally for part 3, UART functionality was experimented with by using outside, physical components. Using an oscilloscope, we were able to familiarize further between the interactions between the board and our equipment. We had changed the original code to send each of the five characters in the word “Hello” to simply “a.’ This was to simplify our experiment. Like programs 1 and 2, Delay and SystemInit methods were added.