Embedded Software Essentials

Project 3: UART and Logger

Zachary Vogel & James Pentz

Professor: Alexander Fosdick

April 10, 2016

Introduction

This project is built to teach us about implementing UART on a microcontroller and using that for logging. To do this, we will write functions to convert numbers to strings, implement the actual UART on the KL25Z, and write another function that implements a circular buffer which can be used for data transfers with the UART. The code for this project can be found at Zach's github, zapv1348 at github. The url is: https://gihub.com/zapv1348/ecen5013_project3.git.

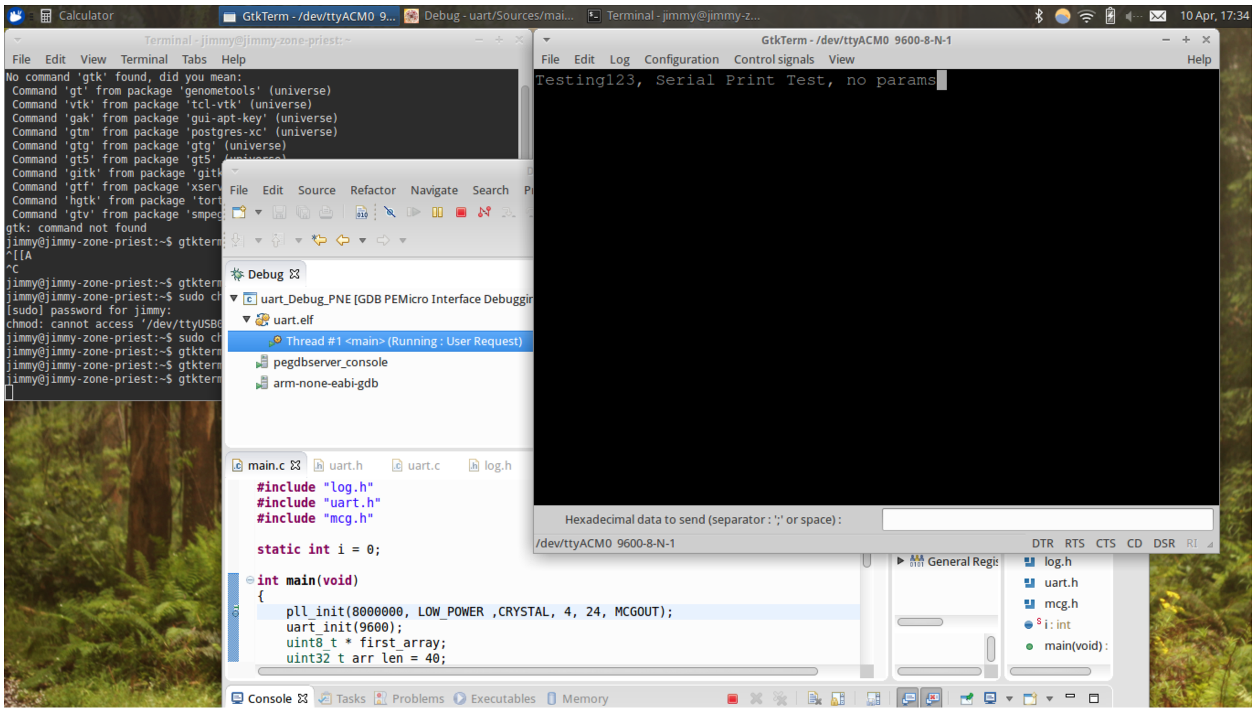
Project Procedure Results

Part 1

The strategy for our itoa was to only add a negative sign for radix 10 numbers. If one needs a negative sign than take the absolute value of the number one wants and throw a negative on the front of the string. Then all one has to do is take the modulo of this value by the radix until it is zero, adding the modded value to the string and dividing the value by the radix every time. This yields a reversed string which we then flip back. ftoa is the same thing as this, but you do it twice. Once for the integer part of a float and once for the decimal part with a dot in between. It is important to note that neither of these functions end the string with a null terminator so one has to be careful. The ftoa should take upwards of 32 total characters and itoa will as well. You can do a unit_test on this by running make unit in the toa directory.

Part 2

In this section of the project we implemented uart communication to a serial terminal. To do this, we first initialized the UART with the proper parity, and data rate. Then we implemented two logging functions, the first which took only a string, and the second which took a string some other parameters. This utilized the functions we wrote in part 1 to convert integers and floating point numbers into ascii strings to be transmitted. The UART connected over the UART to USB converter in the OpenSDA emulator. The requested screenshots can be seen below.

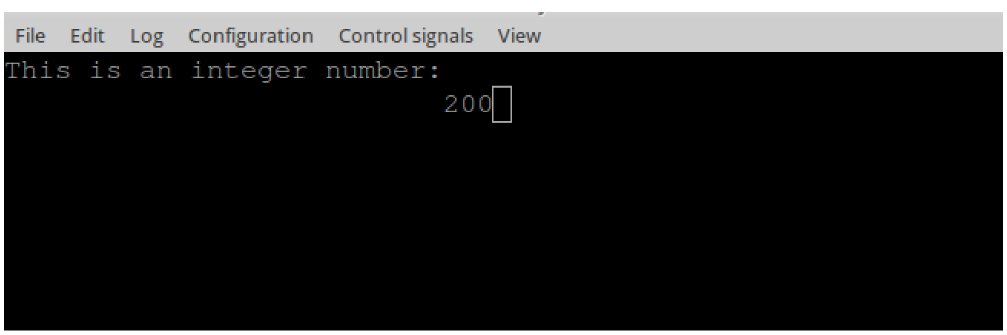

Calculator
GtkTerm - /dev/ttyACM0 9...
Debug - uart/Sources/mai...
Terminal - jimmy@jimmy-z...
10 Apr, 17:34
Terminal - jimmy@jimmy-zone-priest: ~
File Edit View Terminal Tabs Help
GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View Help
File Edit Source Refactor Navigate Search
Debug
uart_Debug_PNE [GDB PEMicro Interface Debuggin
uart.elf
Thread #1 <main> (Running : User Request)
pegdbserver_console
arm-none-eabi-gdb
main.c
uart.h
uart.c
log.h
Hexadecimal data to send (separator : ';' or space) :
/dev/ttyACM0 9600-8-N-1
DTR RTS CTS CD DSR RI
General Regis
log.h
uart.h
mcg.h
i : int
main(void) :
Console
Tasks
Problems
Executables
Memory

```
No command 'gtk' found, did you mean:
 Command 'gt' from package 'genometools' (universe)
 Command 'vtk' from package 'tcl-vtk' (universe)
 Command 'gak' from package 'gui-apt-key' (universe)
 Command 'gtm' from package 'postgres-xc' (universe)
 Command 'gtg' from package 'gtg' (universe)
 Command 'gt5' from package 'gt5'
 Command 'gitk' from package 'gitk
 Command 'gtf' from package 'xserv
 Command 'hgtk' from package 'tort
 Command 'gtv' from package 'smpeg
gtk: command not found
jimmy@jimmy-zone-priest:~$ gtkterm
^[[A
^C
jimmy@jimmy-zone-priest:~$ gtkterm
jimmy@jimmy-zone-priest:~$ sudo ch
[sudo] password for jimmy:
chmod: cannot access '/dev/ttyUSB0
jimmy@jimmy-zone-priest:~$ sudo ch
jimmy@jimmy-zone-priest:~$ gtkterm
jimmy@jimmy-zone-priest:~$ gtkterm
jimmy@jimmy-zone-priest:~$ gtkterm
```

```
Testing123, Serial Print Test, no params
```

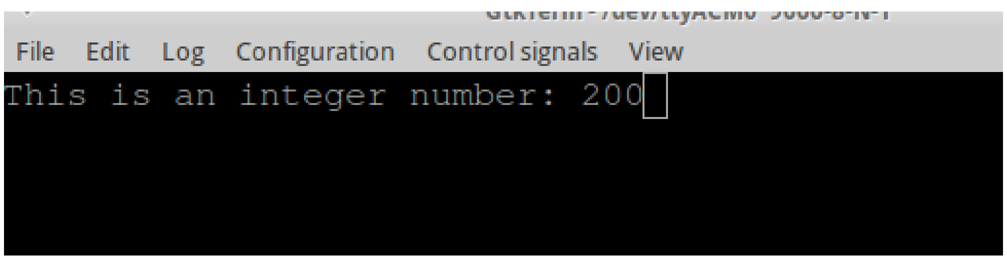
```c
#include "log.h"
#include "uart.h"
#include "mcg.h"

static int i = 0;

int main(void)
{
    pll_init(8000000, LOW_POWER ,CRYSTAL, 4, 24, MCGOUT);
    uart_init(9600);
    uint8_t * first_array;
    uint32_t arr_len = 40;
```

Figure 1: First requested string



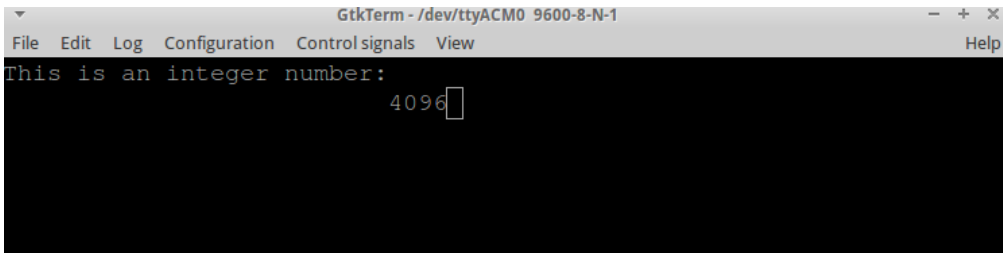
```
This is an integer number:
                               200
```

Figure 2: Unconcatenated string 2



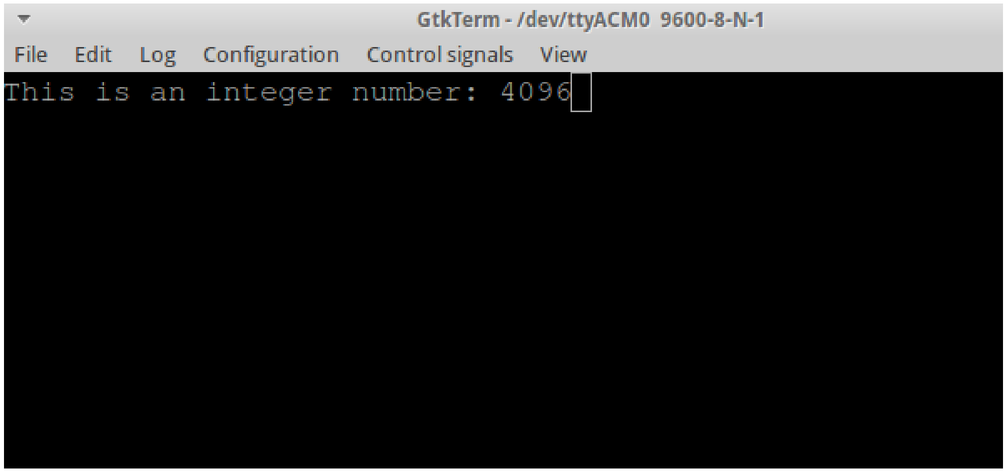
```
This is an integer number: 200
```

Figure 3: concatenated string 2


GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View Help

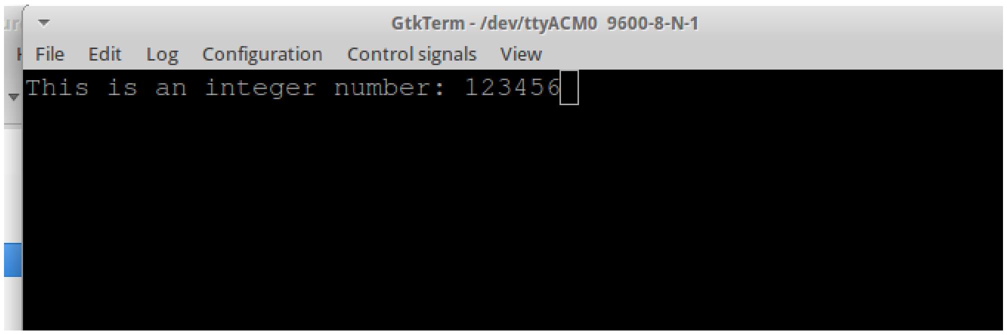
```
This is an integer number:
                           4096
```

Figure 4: unconcatenated string 3


GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View

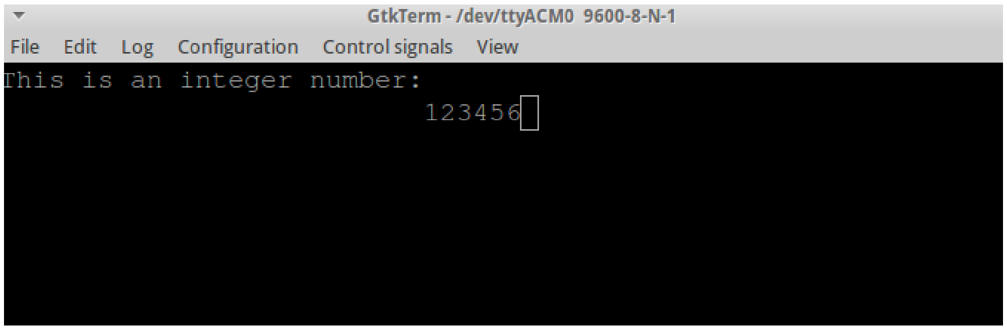
```
This is an integer number: 4096
```

Figure 5: concatenated string 3


GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View

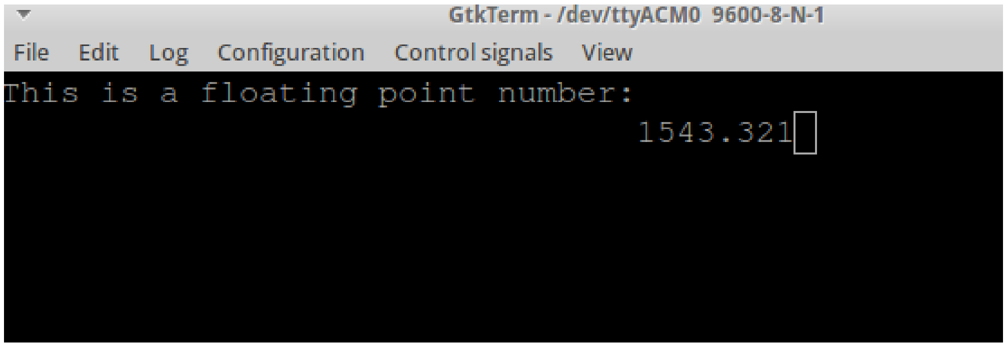
```
This is an integer number: 123456
```

Figure 6: unconcatenated string 4


GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View

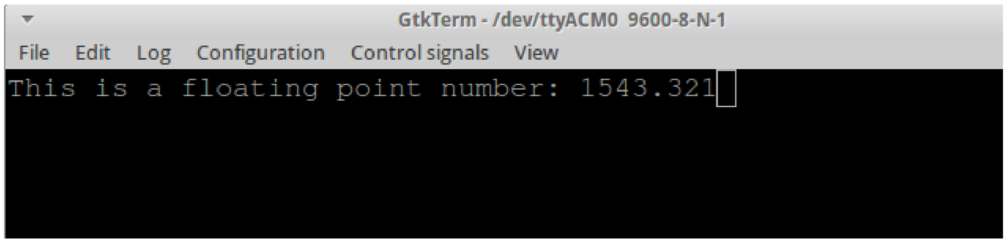
```
This is an integer number:
                           123456
```

Figure 7: concatenated string 4


GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View

```
This is a floating point number:
                                1543.321
```

Figure 8: unconcatenated string 5


GtkTerm - /dev/ttyACM0 9600-8-N-1
File Edit Log Configuration Control signals View

```
This is a floating point number: 1543.321
```

Figure 9: concatenated string 5

Part 3

The strategy here was to initialize the memory to the heap, as well as other values for the struct. Then we create an add and remove function. The important thing here is checking the value type so we know we are moving data around properly. To execute the unit tests pass in the word unit to the make command. The two tests check for everything required, but there is only 2 of them. The first test checks for both fill completely and empty completely as well as that the pointers are incrementing properly. Second checks for the condition where the buffer isn't full, but the head is at the end and you add another term.

Conclusion

This project has been focused on setting up software to be used in conjunction with specific UART hardware built into a microcontroller. First, we explored conversion from number representations to ascii representations. Then we explored initializing a UART and using that UART to log information in part 2. Finally, we developed a ring buffer to organize data coming in and going out of the UART connection. Overall it was quite useful in our pursuit of understanding embedded firmware, it was a project that draws direct contrasts to real world applications.