ECE 331

Homework 5

See course web site for due date

Place your typed homework answers in `vim`. Print single sided with your name using a **mono space font**. No need to restate questions. Fully investigating questions is required for a higher grade. Please use the kernel coding style for all code. Please use your RPi for developing answers. Although code should be written and run on a RPi, it should run on ANY POSIX compliant OS. As always, all code shall be comment, conform to the Linux Kernel Coding Style, and error conditions shall be checked and appropriately handled.

Place ALL code, scripts, and text files (questions 2 to the end of the homework) in gitlab and use the name "`umorse`" for the project name. Add the user "Sheaff" as a developer.

1. Write a perl style regular expression that matches a C style floating point constant. Refer to the ANSI C standard (`n1570` from 2011 `[C11]`) is just fine) for a valid floating point constant. Exclude (no need to consider) hexadecimal floating point representation.
   a) Give the minimum match.
   b) Include your regular expression as your answer for this part.

   No partial credit.

2. Create a text file, by hand, that encodes all Morse code characters, one character per line. Use an asterisk for a dit (one Morse time unit) and three dashes for a dah (worth three Morse time units). Place a space character (one Morse time unit) between every dit or dah and after each Morse character (ease reading and parsing). Example 'A' (dit dah) Morse character

   ```
   A * ---
   ```

   The timing for Morse code follows the above text encoding. Total time for an "A" is 5 Morse time units. Between each character is three Morse time units. Between words is seven time units.

   Write a perl script that converts the ASCII Morse data just created to C source and header files. Each asterisk, space (but not the space in the second column of the file), and dash will represent one bit. Asterisks and dashes will be '1's and spaces '0's. Example for 'A' in binary.

   ```
   0b11101
   ```

   The start the Morse encoding with the LSb. Organize your data. Understand that the C encoding of the Morse data is VARIABLE in length. The output of the perl script will be two files: `encoding.c` and `encoding.h` Encode all 256 C characters (that is all possible byte values).

   Organize your data. Understand that the C encoding of the morse data is VARIABLE in length. The output of the perl script will be two files: encoding.c and encoding.h Encode all 256 C characters (that is all possible byte values).

3. Write a C program that accepts a single valid input text string from the command line and print the Morse encoding. For this encoding, print a "*" for every "1" and "_" for every "0". Example for "BYE"

```
***_*_*_*___***_*_***_***___*
B        ^  Y              ^  E
         |                 |
         |                 - Three Morse time units
         - Three Morse time units between each letter.
```

   Remember that seven Morse time units appear between each word.

4. Write a complete Makefile to compile your code.

5. Modify your C program to include a premable. The preamble is always 0b0100.

6. Modify your C program to append a "0" and an eight bit checksum to the Morse code bit stream. For the checksum, count the number of "1" bits including the single one bit in the preamble. Take the one's complement after summing all the bits. Print all eight bits starting with the LSb of the checksum just like it was a letter. Be sure to do all eight bits. Now "BYE" becomes

```
__*_***_*_*_*___***_*_***_***___*_*_**_***
^   B        ^  Y              ^  E^^-> 8 bit checksum
|            |                 |   |
|            |                 |   - Additional "0"
|            |                 - Three Morse time units
|            - Three Morse time units between each letter.
- Preamble (Remember it's LSb first)
```

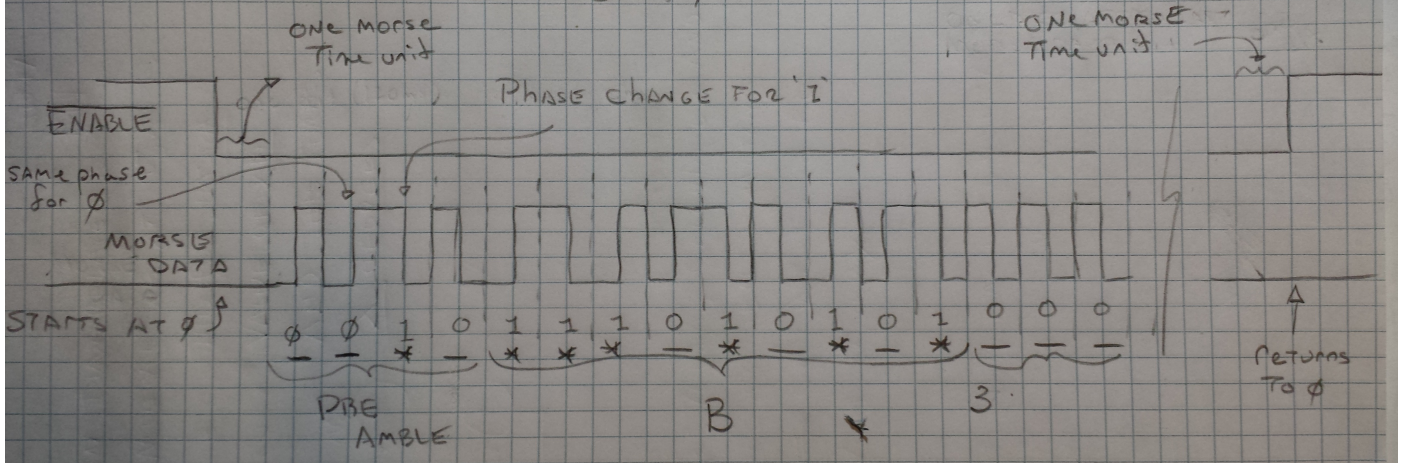
   Message format:

Name	Preamble	Morse Message	Zero	Checksum
Bits	4	At least 1	1	8
Value	0b0100	Varies...	0	Varies..

7. Modify your C program to control output pins 4 and 17 on the RPi using the `sysfs` interface. Pin 4 will be an enable output and pin 17 will be the Morse code data BPSK encoded. Enable is active low. Zero is enabled. One is disabled. For one Morse time unit before sending Morse data, bring the enable low. One Morse time unit after the last Morse data bit, bring enable high. BPSK encodes data in signal phase. A zero data bit means keep the phase the same as the previous bit. A one bit means to change the phase by 180 degrees. Each bit will take on Morse time unit. Begin and end with the Morse data line low. Example encoding for the start, preamble, and a 'B' character follows on the next page.

For your answers to questions 2 through 7, include your final source code (but not `encoding.[ch]`), scripts and Makefile.

MORSE ENCODING


ONE Morse Time unit
Phase change for '1'
ONE MORSE Time unit
ENABLE
SAME phase for Ø
MORSE DATA
STARTS AT Ø
Ø Ø 1 0 1 1 1 0 1 0 1 0 1 0 0 0
PRE AMBLE
B
3
returns To Ø