Program Development Cycle worksheet

*for C-Language program design*

A general C Language CodeWarrior program outline

Name: George Landis

Lab: Lab\_03

Class: ECET 30903

Date:1/15/2019 Due:1/21/2019 11:59 PM

For each assignment you are to use this outline to document the six phases of the program development cycle. Your documentation must be complete and professional, easy to read and include all steps of the program development cycle as in Lecture/Lab

This process should be followed each time you start an assignment. Do not write your programs before you have completed and been checked off for this process.

1. Understand the Program’s Purpose
   1. State the problem/purpose:

The problem/purpose of this lab is creating a physical way of getting a keypad to understand how to be pressed and learning the coding practice for a basic 4 by 4 Matrix Keypad and using parallel arrays. As well as integrating the simulation to a real keypress.

* 1. Then Identify
     + Inputs: List the hardware input(s) needed?
       - A Keypress on the 4x4 Matrix Keypad
     + Process: Simples steps needed to preform program logic
       - Find that there is a keypress
       - Go through the functions to find exactly which key if any have been pressed
       - Display out on the LEDs
     + Outputs: List the hardware output(s) needed?
       - Outputs will be the dedicated from the Keypress onto the LEDs

1. Design/Assign the hardware configuration. (For embedded system design)
   1. Create a table to identify and organize I/O hardware configuration assignments. Make sure to only include the I/Os the program needs.
      * What hardware I/Os are needed? Identify each I/O by hardware name.

4x4 Matrix Keypad & 8-bit LEDs

Example: 8-bit dip switches, 8-bit LEDs, stepper motor …..

* + - * Assign I/O hardware pin/port names and memory mapped Port address: Note: this is dependent on Microcontroller and trainer board used

An entire list of I/O port names and associated addresses is provided in Resources. Example: MC9S12XEP100 chip

4x4 Matrix Keypad are controlled by PORTA at bit 0-7

8-bit LEDs are controlled by PORTC at memory mapped address 0x04

8-bit Dip switches are controlled by PORTB at memory mapped address 0x01

Stepper motor is controlled by PORTD at memory mapped (3-0) address 0x05

* + - * Set the I/O’s data directional register accordingly. Is the I/O used as an input or an output?

Keypad is used as inputs: since all I/Os default as inputs you do not need to set the corresponding Data directional register.

LEDs are used as outputs: 0-7 bits of PORTC are used for the LEDs. Meaning we need to set the bits as outputs DDRC =0xFF;

Example:

Dip switches are used as inputs: since all I/Os default as inputs you do not need to set the corresponding Data directional register.

LEDs are used as outputs: all 0-7 bits of PORTC are used to control the LEDs, so, set these bits as outputs….DDRC = 0xFF;

Stepper motor is used as a output: since bits 3-0 of PORTD control the stepper motor operation, then set these bits as outputs. DDRD |= 0x0F;

Refer to the section at the end of this document called: **When designing a program solutions ask the following:**

Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identify I/Os hardware configuration assignments** | | | | |
| **I/Os Needed** | **Port name** | **Port address** | **purpose** | **Initial setting for 8-bit Port’s DDRx** |
| 4x4 Matrix Keypad | PORTA | 0x00 | input | Not needed since it is an Input. |
| All 8 LEDs | PORTC | 0x04 | output | DDRC = 0xFF (sets bits 0-7) |

1. Design the Program Logic:

Break down each task that the program must perform into a series of logical steps.

Example:

Check a bit-state on the dip switches (is the bit on or off?) Light the LEDs

Turn the stepper Motor

* 1. Develop an algorithm pseudo code for your program:

Pseudo Code Here

/\*

* + - * Setting a Keypad press in the main function
      * Calling the DetectedNumericKey & CheckAsciid functions.
      * Each function is in the Keypad.c file that will then work through
      * The DetectedNumericKey function will grab the detectedKeyPress array and find when a key has been pressed. Search for that keypress numerical array number in the numericVal. If cannot find returns 0xFF.
      * The CheckAscii function will do the same thing by grabbing from the detectedKeyPress array and find when a key has been pressed. Search for that keypress within its ascii array in the asciiVal. If it can not find results it too will return 0xFF.
      * The Final result will be passed to a Variable as well as a passing variable going between blocks.

\*/

Algorithm: a set of well-defined, logical steps that must be taken to perform a task. This is not a copy of code you have already written. This is the logic design you think your program needs to perform the identified tasks. It is not a copy of code

* 1. Flowchart the program’s algorithm:

Flowchart: a diagram that graphically depicts the steps of an algorithm, where specific symbols represent each step in the algorithm.

* Identify any function(s) if needed for tasks identified. Will the function receive any arguments, will the function return a value, what variable(s) will the function need to perform its task? Use table as exampled on page 3.
* Identify variable(s). Determine if the variables are to be global, local, uninitialized, initialized and/or constants. Use tables as exampled on page 3.

For In-Lab credit:

At this point you are ask to have an Instructor or TA check your Program Development Cycle worksheet. You will continue to work on the remaining three design cycle tasks in CodeWarrior and submit the entire finished document with your lab assignment.

The remaining steps are done in CodeWarrior. Only update steps 1-3 if you find your original design algorithm(s) and flowchart needed correction due to logic errors.

1. Write the Code: Only after you have designed algorithms and flowchart for the program’s tasks (steps 1-3) do you start writing code.

Do not try to write the program all at once.

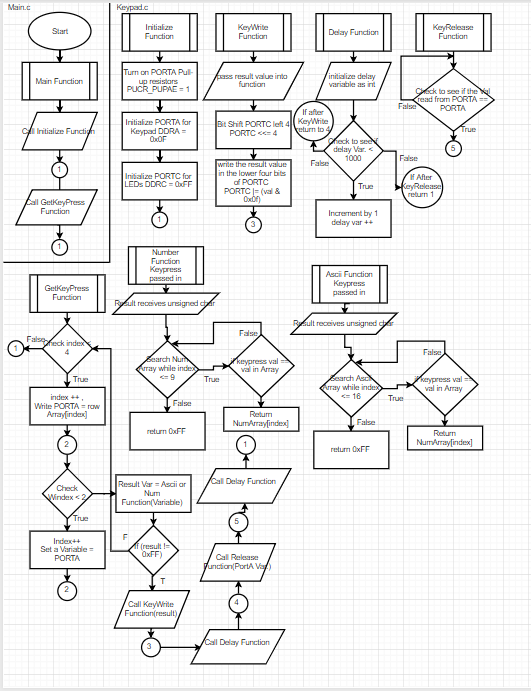
* Write in small pieces
* correct syntax errors as they occur (step 5) and
* run the program and test for logic errors (step 6).

1. Correct Syntax errors:

* A syntax error occurs if the programmer violates the rules of the programming language used. CodeWarrior has rules that must be followed, and if not, then the program cannot be compiled or executed.
* You must read the assigned reading to understand and learn how the variables, constants and code are organized in code sections. And know how to use the keywords, code statements, memory addresses and logic execution for the C-Language and CodeWarrior IDE.

1. Test the Program and Correct Logic Errors:

* A logic error is a mistake that does not prevent the program from running but causes it to produce incorrect results.
* This process is called “debugging” or “debug the code”: find and correct the logic errors.
* Sometimes the logic error is due to program design



# When designing a program solution ask the following:

* What I/Os are needed? Identify each I/O by hardware name. Example: dip switches, LEDs, stepper motor …..
* By what Port is the I/O controlled? Identify each I/O port that will be used. Example:

Stepper motor is controlled by PORTD Dip switches are controlled by PORTB

Note: An entire list of I/O port names and associated addresses is provided in resources

* Is the I/O used as an input or an output? Set the I/Os data directional register accordingly Example:

Dip switches are used as inputs: since all I/Os default as inputs you do not need to set the corresponding Data directional register.

LEDs are used as outputs: all 0-7 bits of PORTB are used to control the LEDs, so, set these bits as outputs….DDRB = 0xFF;

Stepper motor is used as a output: since bits 3-0 of PORTD control the stepper motor operation, then set these bits as outputs.. DDRD |= 0x0F;

Example I/Os table:

* What are the processes? Define them. Example:

Check a bit-state on the dip switches (is the bit on or off?) Turn the stepper Motor

Light the LEDs

* What functions will you define to complete the processes identified? Identify the variables needed as well. Determine if the variables are to be global, local, uninitialized, initialized
  1. Be sure to include a prototype for each function you declare.
  2. Be sure program components, such as variable declarations and instruction code, are placed as outlined above. CodeWarrior is very particular with regard to the order by which uninitialized variables, initialized variables, functions, function prototypes, and instruction code is placed within a program. It is crucial that your program follows the outline below and observes the proper order to variables and instructions.

# What functions will you define? Example function(s) table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Identify function(s)** | | | |
| **Return data type** | **Function name** | **Parameter list** | **purpose** |
| Unsingned char | DetectedNumericKey | (unsigned char) | Receives an unsigned character from the detectedKeyPress Array, looks for a numeric digit in another array from 0-9 if not found returns 0xFF |
| unsigned char | CheckAscii | (unsigned char) | Receives an unsigned character from the detectedKeyPress Array, looks for an Ascii char listed in 0-9,A-D,\*,or #, returns one of those or if not found 0xFF |

**What variables will you declare? Example variable(s) table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identify variable(s)** | | | |
| **data type** | **Variable name** | **Initial value** | **purpose** |
| Const unsigned char | detectedKeyPress | [ ] | Holding all Hex values for the Keypad as an array |
| Const unsigned char | numericVal | [ ] | Holding all numeric values for the keypad press 0-9 as an array |
| Const unsigned char | asciiVal | [ ] | Holding all alphanumeric values via ascii for the keypad press 0-9,A-D,\*. and #. As an array. |
| Unsinged char | finalVal | 0 | Holds the final value to be displayed on the 8-bit LEDs |
| Unsinged char | passVal | 0 | Holds the value passed between arrays. |

Make sure to view the following page: **CodeWarrior program shell code organization that follows.**

The following C language CodeWarrior program shell provides an outline for all C language programs created for ECET30903.

In the main.c file put the following commented header at top of file.

// Programmer(s) Name: My Name

// Title of Program: Lab ##

// date: xx/xx/xxx

// class: ECET 30903

// due date: xx/xx/xxxx

// Brief explanation of program, one to three lines

// the following two lines are the default system included header files #include <hidef.h> /\* common defines and macros \*/

#include "derivative.h" /\* derivative-specific definitions \*/

// any additional files added to the program solution should follow

/\* Function Prototypes go here:

* only when no additional .h or .c files are requested
* otherwise
* always put Function Prototypes in a separate .h file \*/

//uninitialized Global variables go next

//initialized Global variables go after uninitialized variables

// main function void main(void)

{ /\* put your own code here \*/

// declare un-initialized local variables

// declare initialized local variables next

/\* call functions and write any code statements that require

\* a one-time process: such as initialize data, I/O ports….

\*/

for(;;)

{

// call functions that require a continuous process:

// example: checking a dip switch bit-state

\_FEED\_COP(); /\* feeds the dog \*/

} /\* loop forever \*/

/\* please make sure that you never leave main \*/

} // end of main function

/\* write your function declarations outside of the void main(void){ } function.

For this class: when instructed place additional functions after the main function. Otherwise you will place your additional functions in separate .c file(s).

(Note: if you have declared all your function prototypes, then CodeWarrior will allow functions written

/placed before or after the main function, but we will only put them after main() or in a separate file )

\*/

// end of main.c file

Note!:

When required in lab instructions to have additional files other than the main.c file you will have a separate .h file (called a header file) for you Function prototypes and you will add additional .c file(s) to your project files. Specific instructions on how this is done will be given in lecture and lab instructions.