Design Assignment 1 – Embedded ‘C’ Programming Basics

Josh Tobia

Linh Ngo

Brian Dawson

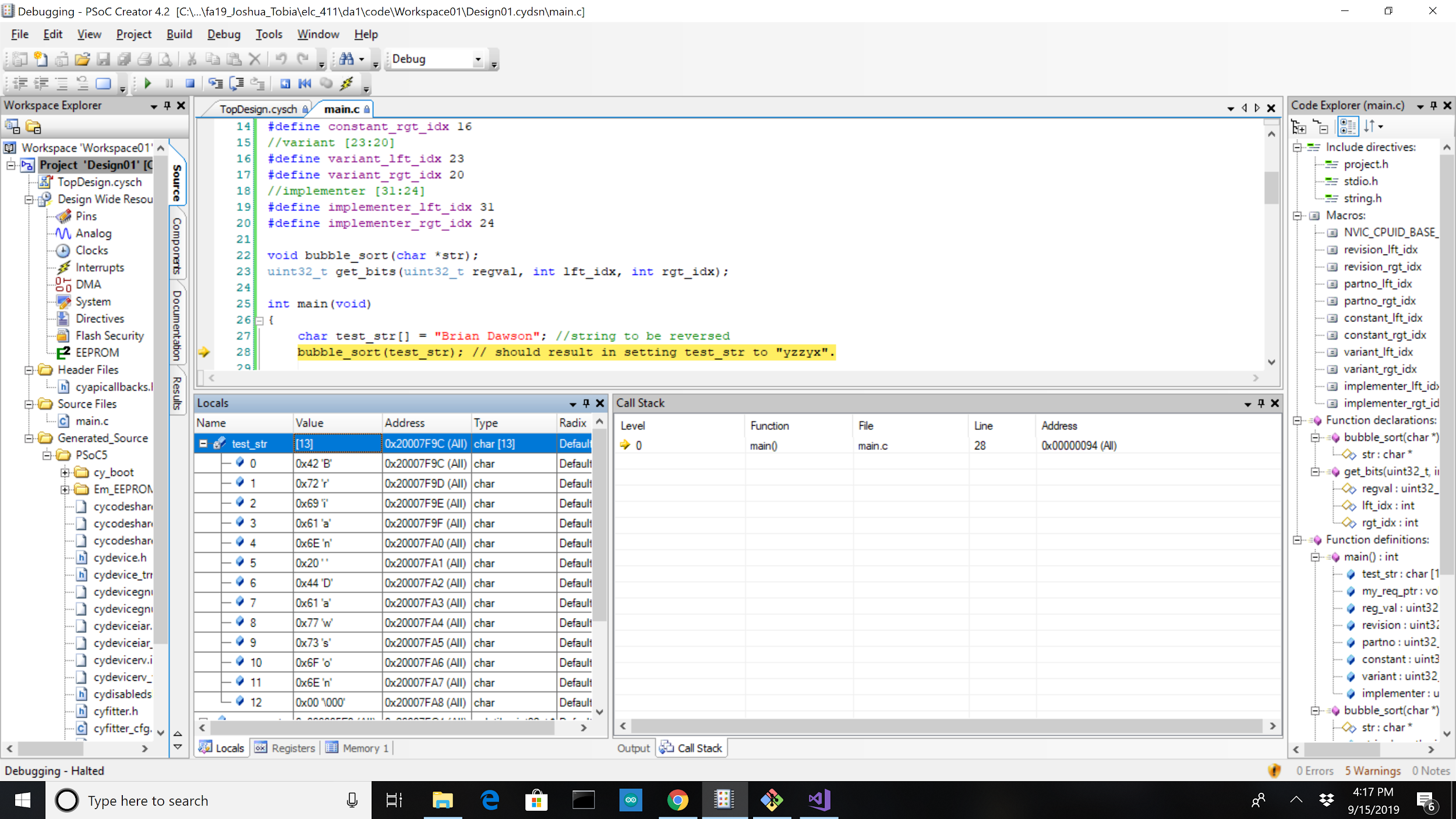
ELC 411 - Embedded Systems

The College of New Jersey

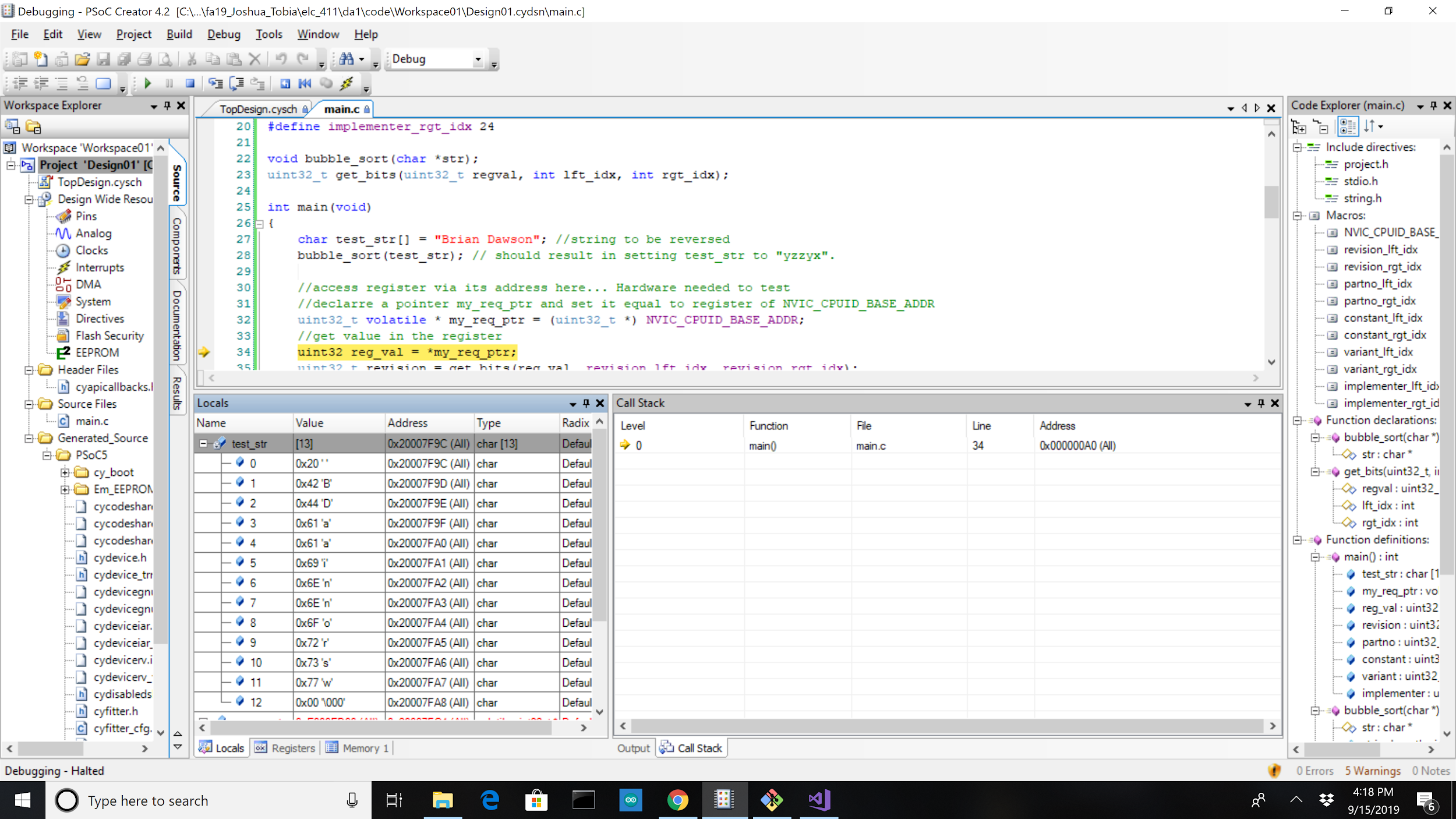
Department of Electrical and Computer Engineering (ECE)



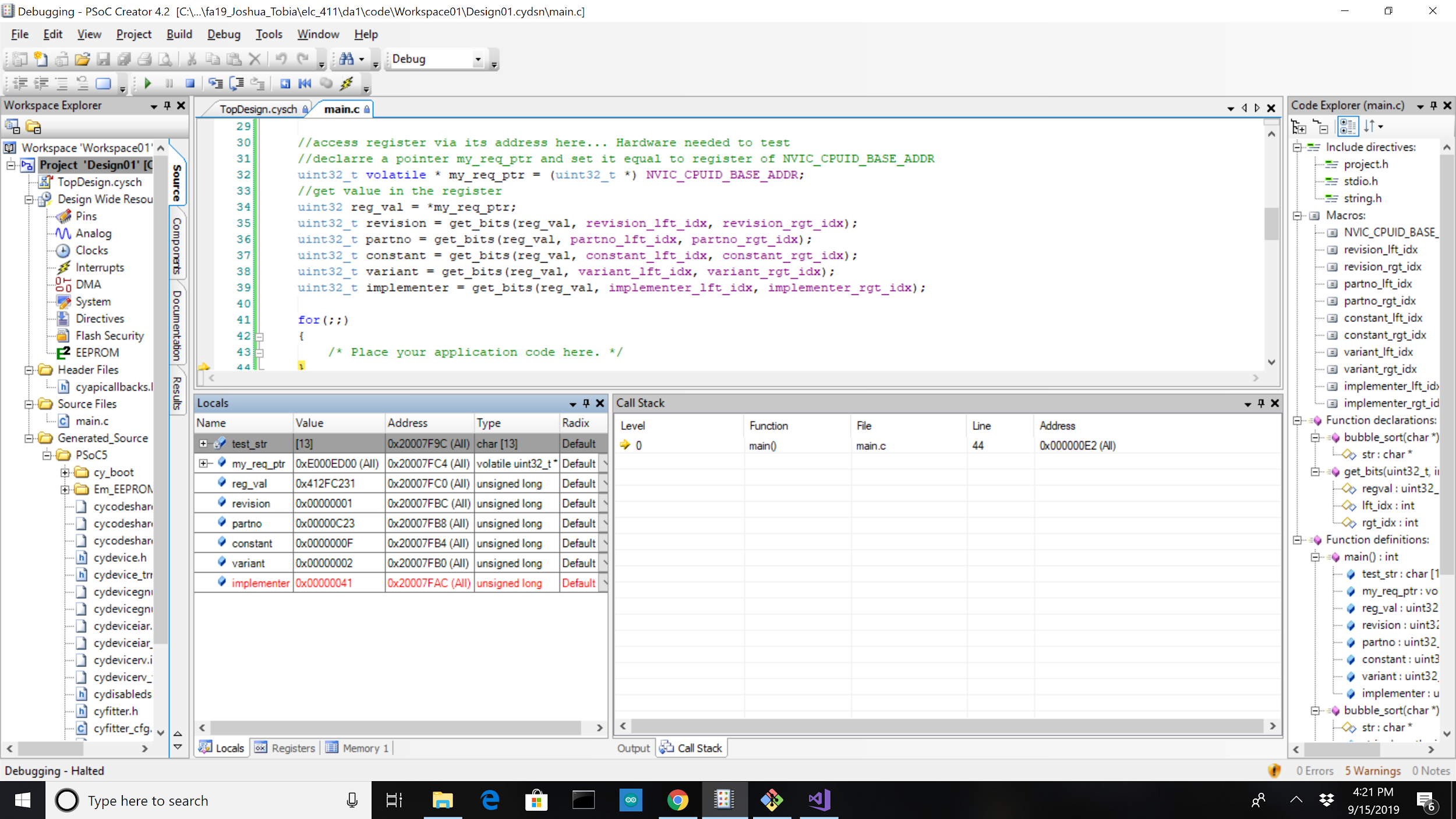
**Date Submitted: September 17, 2019**



*Figure 1: Memory Address and Character Value of test\_str Before Bubble Sort Function is Called*



*Figure 2: Memory Address and Character Value of test\_str After Bubble Sort Function is Called*



*Figure 3: Values of Sub-Fields from Register Address through Watch Window in PSoC Creator*

1. In “C,” how would you extract the value of a register field occupying bits [5:3] from a register stored in a variable regval?

You would need to right shift the value stored in regval by 3 positions and then bitwise AND the resulting value from that operation with 111 base 2.

1. How would you modify regval to insert a new value into these bits, based on the integer value stored in a variable fieldval, where we know that the value in fieldval is in the range 0...7. Note -- you need to insert the desired bits without changing any other bits in the register.

To insert fieldval into regval there are three main steps. The first step is to AND with 0’s the bits 5 to 3 in regval. The next step is to shift the bits in fieldval over by three to the left so that they line up with the values in regval. The final step is to OR regval with fieldval.

1. Why does the space character (s) get sorted to the start of the array?

All the characters get sorted based on their corresponding ASCII value; ASCII assigns characters with a numerical value. The space character has the lowest ASCII value in comparison to the rest of the keyboard characters. The space character has a value of 32 decimal, while uppercase and lowercase alphabet letters have a value between 65 and 122 decimal.

1. Briefly explain the function of the register PRT6\_DR.

This is a port data output register; the register is used to set the output data state for the corresponding general purpose input/output (GPIO) port. The data written to this register specifies the high or low state for the GPIO pin at each bit location of the selected port.

C Code :

… Your main() procedure code, strictly formatted using: (4 points)

o spaces, not TAB characters for indenting

o 4 spaces for each level of indentation

o fully commented

o **monospaced font (e.g. Courier New)**

#include "project.h"

#include <stdio.h>

#include <string.h>

#define NVIC\_CPUID\_BASE\_ADDR 0xe000ed00

//revision [3:0]

#define revision\_lft\_idx 3

#define revision\_rgt\_idx 0

//partno [15:4]

#define partno\_lft\_idx 15

#define partno\_rgt\_idx 4

//constant [19:16]

#define constant\_lft\_idx 19

#define constant\_rgt\_idx 16

//variant [23:20]

#define variant\_lft\_idx 23

#define variant\_rgt\_idx 20

//implementer [31:24]

#define implementer\_lft\_idx 31

#define implementer\_rgt\_idx 24

void bubble\_sort(char \*str);

uint32\_t get\_bits(uint32\_t regval, int lft\_idx, int rgt\_idx);

int main(void)

{

char test\_str[] = "Brian Dawson"; //string to be reversed

bubble\_sort(test\_str);

//access register via its address here... Hardware needed to test

//declare a pointer my\_req\_ptr and set it equal to register of NVIC\_CPUID\_BASE\_ADDR

uint32\_t volatile \* my\_req\_ptr = (uint32\_t \*) NVIC\_CPUID\_BASE\_ADDR;

//get value in the register

uint32 reg\_val = \*my\_req\_ptr;

uint32\_t revision = get\_bits(reg\_val, revision\_lft\_idx, revision\_rgt\_idx);

uint32\_t partno = get\_bits(reg\_val, partno\_lft\_idx, partno\_rgt\_idx);

uint32\_t constant = get\_bits(reg\_val, constant\_lft\_idx, constant\_rgt\_idx);

uint32\_t variant = get\_bits(reg\_val, variant\_lft\_idx, variant\_rgt\_idx);

uint32\_t implementer = get\_bits(reg\_val, implementer\_lft\_idx, implementer\_rgt\_idx);

for(;;)

{

/\* Place your application code here. \*/

}

}

void bubble\_sort(char \*str)

{

int stringLength,i,k;

for (stringLength = 0; str[stringLength] != '\0'; stringLength++) {}

//calculates string length

char \*currentChar, \*nextChar, temp;

currentChar = str; //copying str pointer to currentChar

nextChar = str; //copying str pointer to nextChar

nextChar++; //moving nextChar pointer to second character in str

for (k = 1; k < stringLength; k++)

//defines the number of iterations of bubble sort. Will be stringLength-1 iterations

{

for (i = 0; i < stringLength - k; i++)

//i< stringLength - k because each iteration puts the largest value last. This means

//one less comparison is needed each iteration

{

if (str[i] > str[i + 1])

//if the currentChar is greater than the char right after it then switch them

{

temp = \*nextChar; //save nextChar into temp

\*nextChar = \*currentChar; //set nextChar = currentChar

\*currentChar = temp; //make currentChar equal to temp(original nextChar)

}//end if (str[i] > str[i + 1])

nextChar++; //moving nextChar to right one character

currentChar++; //moving current char to the right one character

}//end for (i = 0; i < stringLength; i++)

for (i = 0; i < stringLength - k; i++) //same for loop as above to reset pointer locations

{

nextChar--; //moving nextChar back to second character

currentChar--; //moving current char back to first character

}//end for (i = 0; i < stringLength - k; i++)

}//end for (k = 0; k < stringLength - 1; k++)

}//end void bubble\_sort(char \*str)

uint32\_t get\_bits(uint32\_t regval, int lft\_idx, int rgt\_idx)

{

uint32\_t bufferVal = 0; //bufferVal will be used to create a binary value of 111... with

//lft\_idx many 1s

int i;

regval = regval >> rgt\_idx; //shifts right by rgt\_idx bits

int nbits = lft\_idx - rgt\_idx + 1;

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

{

bufferVal = (bufferVal << 1) + 1; //shift right by 1 bit then add 1 (add an

// additional 1)

}//end for (i = 0; i < nbits; i++)

regval = bufferVal & regval; //cuts off all of the bits past the lft\_idx point

return(regval);

}

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Expected** | **Pts. Available** | **Deduction** |
| **Title Page** |  | **1** |  |
|  |  |  |  |
| Present observed results: |  |  |  |
| imlementer | 0x41 | 0.4 |  |
| variant | 0x2 | 0.4 |  |
| partno | 0xC23 | 0.4 |  |
| revision | 0x1 | 0.4 |  |
| reg\_val | 0x412FC231 | 0.2 |  |
| string prior to sort | varies | 0.4 |  |
| string after sort | sorted string | 0.8 |  |
| **SUBTOTAL** |  | **3** | **0** |
|  |  |  |  |
| Questions: |  |  |  |
| 1) Extract reg. field | (regval & 0x38) >> 3 | 0.5 |  |
| 2) Modify regval | (regval & ~0x38)|(fieldval <<3) | 0.5 |  |
| 3) Why space at start? | ASCII for space is 0x20, less than alphanums | 0.5 |  |
| 4) PRT6\_DR | Data written specifies state for pin at each bit of port 6. | 0.5 |  |
| **SUBTOTAL** |  | **2** | **0** |
|  |  |  |  |
| **main.c file, formatted and commented** |  | **4** | **0.3** |
|  |  |  |  |
| **TOTAL** |  | **10** | **9.7** |