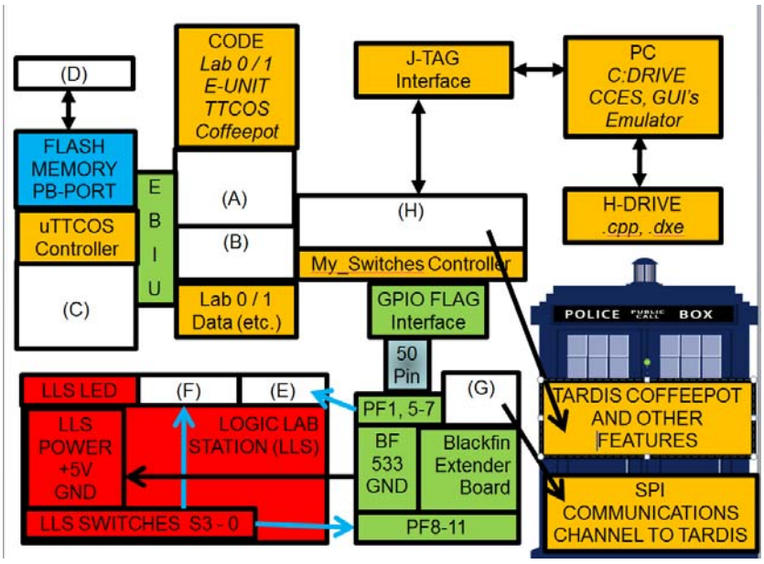
ENCM511 MIDTERM 2014 – KEY

Fig. 1



LISTING 1

```
1  .section L1_data
2     .byte2  _tardis_door = 1;
  // Some comments here
6  .section program
7  .global _Tardis_Music;
8_Tardis_Music:
  // bits of code here
10  P0.L = lo(SPI_TDBR); P0.H ......
  // bits of code here
15   Call  _uTTCOS_ReadLED;
  // bits of code here
20  P1.L = lo(FIO_DIR); P1.H  .....
  // bits of code here
25 P4.L = lo(FIO_INEN); P4.H .....
  // bits of code here
30  P2.L = lo(TIMER1_ACKNOWLEDGE) ;  P2.H ...
   //  bits of code here
35     RTS;
36 _Tardis_Music.END:
  // Some comments here
40   .section sdram_data
41    .byte4  _tardis_drive = ON;
```

Q1) Labels from various parts of the Blackfin embedded system controlling the TARDIS have been blanked out and replaced by letters. Provide the names of the embedded system part in the second column of the table below, and which line, or lines, of code in the listing in the third column is most likely to be controlling that part of the embedded system. **Item H has been completed for you as an example**

Missing Label	Name of the embedded system part (1 / 2 mark)	Line of code in the listing most likely used to control that part of the embedded system. (1 mark)
(A)	*Blackfin L1_CODE memory*	*lines 6 to 36*
(B)	*Blackfin L1_DATA memory*	*lines 1 and 2*
(C)	*SDRAM external memory*	*lines 40 to 41 – **How many added line 42 for a bonus?***
(D)	*LEDS attached to external memory*	*line 15*
(E)	*LLS LEDS 3-0 or equivalent*	*line 20 – controlled by PF lines in output*
(F)	*LLS LEDs 15-11 or equivalent*	***NONE OF THE CODE – You CAN'T read or write to these LEDS as they are not connected to a Blackfin***
(G)	*SPI interface*	*line 10 – SPI transmit buffer (Quiz 1)*
(H)	**Blackfin Microprocessor Core**	**I suppose – everything – lines 1 to 41**

Q2) (A) Explain the difference between a DESIGN MISTAKE, a DESIGN DEFECT and a DESIGN ERROR (2 marks)
(B) Which of a DESIGN MISTAKE, a DESIGN DEFECT and a DESIGN ERROR is likely to be the most expensive to fix for a company developing a product. **Explain why that is likely to be the most expensive** (2 marks)
If you mentioned anything related to code defects or errors then NO MARKS in either question**.**
A design error is a design mistake which you recognize and fix BEFORE moving onto the next phase (testing or coding). A design defect is a mistake made during the design that you don't recognize until much later in the project possibly causing you to throw away a lot of working code or abandon the whole project.

(C) You have considerable experience through Lab. 0, Lab. 1, Assignment 1 and Assignment 2. Consider the Dr. Who Coffeepot listing on the previous page. Provide 2 frequent and expensive CODE DEFECTs and 2 frequent and expensive CODE ERRORS that you would expect to occur within this code based on your own experience or through observing your laboratory partner or other members of the class (or my lab. handouts :-))

Discuss what is needed to fix them. (2.0 marks each)
FREQUENT AND EXPENSIVE CODE DEFECT FOR YOU, FREQUENT AND EXPENSIVE CODE DEFECT FOR YOU
FREQUENT AND EXPENSIVE CODE ERROR FOR YOU, FREQUENT AND EXPENSIVE CODE ERROR FOR YOU
As long as you explain the problem and it is correctly identified as a defect or error and there is a reasonable (1 or 2 line) explanation – then you get the marks. If incorrectly identified as a defect or error then no marks

The following shows the structure of the key parts of the TARDIS COFFEEPOT control device.

```
typedef struct TARDIS_CDR_STRUCTURE  {          // TARDIS_COFFEEPOT_DEVICE_REGISTER_STRUCTURE
    unsigned long long int TARDIS DRIVE_register          // A 64 bit pattern controlling general TARDIS switches
    unsigned char heaterRegister;                         // A VALUE (0 to 255, 0x00 --> 0xFF)
    unsigned char waterInFlowRegister;                    // A VALUE (0 to 255, 0x00 --> 0xFF)
    unsigned short int controlRegister;           // A BIT pattern controlling coffeepot switches
} TARDIS_COFFEEPOT_DEVICE;
```

This structure is very similar to the coffeepot structure used in Assignment 1 and Assignment 2 except for the following

1) The INIT_AND_STAY_POWERED_ON_BIT still activates the device but is **now bit 3** of the TARDIS control register.
2) **There is NO coffee-pot ready bit in the TARDIS control register**. Instead, you must turn on BIT 42 of the TARDIS DRIVE register. This takes the TARDIS 42 milli-seconds into the future so that the coffeepot is immediately ready to be used. This bit automatically becomes 0 once the TARDIS re-appears in the future.
3) The WATER_ON_BIT is **now bit 8** of the TARDIS control register
4) There is a unsigned long long int TARDIS DRIVE register. This 64 bit register is needed as there are so many things to control inside the TARDIS. Careful – BIT 63 is reset to the ON position when the TARDIS embedded system is activated. This controls the "swimming pool" which is an important feature of a number of Dr. Who stories. Turning BIT 63 off causes all the water to flow out of the swimming pool into the TARDIS control room and the TARDIS will be destroyed (will never re-materialize).

Q3A) What is the hex bit pattern to turn on **BIT 42 of the unsigned long long int TARDIS DRIVE register?** 1 mark
The bottom 32 bits would be 0000 0000 so we need 0x200 0000 0000

Q3 B) The pseudo- code describes how to control the TARDIS coffeepot. Translate the pseudo code design into Blackfin assembly code following the coding conventions of tis course 10 marks

Blackfin assembly code	TARDIS COFFEE-POT DESIGN
.section L1_data; *.byte4 water = 0;* ***// Hints from Q1***	*volatile int waterFlowRate_Used = 0;*
.section program; *.global _TWC_NM;* *_TWC_NM:*	*unsigned char TARDIS_Water_Control(* *TARDIS_COFFEEPOT_DEVICE * pt (R0) ,* *unsigned char required_WaterLevel) (R1) {*
P0_pt = INPAR_R0; ***_TWC_NM:*** *ControlRegister_R2 = W[P0 + 18]:* *R3 = 0x100 | 0x008; // Bits 8 and 3* *ControlRegister_R2 = ControlRegister_R2 | R3;* *W[P0 + 18] = ControlRegister_R2;*	// Activate the device by *setting the INIT_AND_STAY_POWERED_ON_BIT and turning on the water power*
// Tricky – how handle 64-bits *// Answer from Assignment 1 – using time* *// Use 32 bit operation on top 32 bits of register* *DRIVE_R2 = [P0 + 0]; // Top 32 bits* *R3 = 0x200;* ***// OR to save pool water bit 63*** *DRIVE_R2 = DRIVE_R2 | R3;* *[P0 + 0] = DRIVE_R2;*	// Set BIT 42 of the TARDIS DRIVE register. // After the TARDIS jumps into the future, the coffeepot device will automatically be ready for use // This assumes Blackfin is BIG-ENDIAN
waterRate_R2 = 25; // (255 / 10) *B[P0 + 6] = waterRate_R2;* *P1.L = lo(water); P1,H = ...* *[P1] = waterRateR2;*	// Set the water flow rate *waterFlowRate_Used* close to (but not above) 1/10 of maximum water flow rate **I decide to save waterFlowRate here and not later**
// Prepare to call C++ code *[--SP] = R7;* *requiredLevel_R7 = requiredLevel_inpar2_R1;* *LINK 20;* *.extern _ReadWaterLevel_NM1;* *LOOP:*	// While the *coffeepot water level is less than the required_WaterLevel* { // You will need to call // *unsigned char ReadWaterLevel(void);*
CALL _ReadWaterLevel_NM1; *CC = return_R0 < requiredLevel_R7;*	// Do nothing }
IF CC JUMP LOOP; // Not enough water	// *Store the waterFlowRate_Used* for later analysis
TURN OFF THE WATER SO IT DOES NOT OVERFLOW *Unlink;* *R7 = [SP++]*	// There were some instructions here – but they got erased and we can't read them. **STATE WHAT THE ERASED COMMENTS MUST HAVE SAID TO MAKE THE CODE WORK CORRECTLY. NO ASM NEEDED.**
RTS; *_TWC_NM.END:*	}

It is getting near the end of the 2041 Dr. Who series so the current companion is about to suffer the fate of previous companions

A) Mind manipulated and unable to remember the horrible experience that will occur in the last episode

B) Marooned on some planet in inter-dimensional space, never to hear a Dr. Smith joke again – etc.

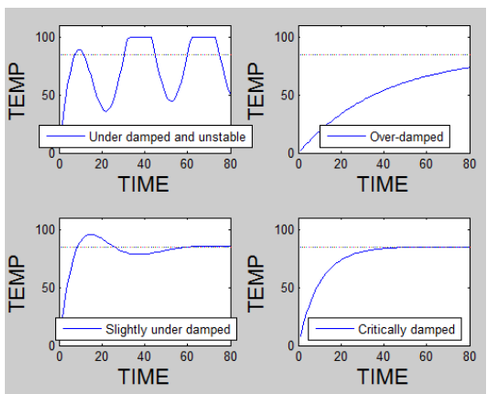
You, a successful ENCM511 student, have been brought from 2014 to be interviewed to possible star as the 2042 companion. **Q4 and Q5 are the interview questions. Make sure you read and understand the Q5 interview question before attempting to answer the Q4 interview question.**

Q4A) You were transported on 31st October, 2014 Halloween, dressed in a " ????????" costume and will be returned in time to take this midterm. Choose an important fictional character from your culture to be for the rest of the interview. Explain how your character fits into the Dr. Who theme. **I need a story to use in Prelab Quiz 3 and the Final.**

I always think about that captain from the Arthur C Clarke story – Rendezvous with Rama 2 marks

Q4B) A previous interviewee made four attempts to control the coffeepot temperature level at 85 C. Having successfully completed Assignment 2 and read my email hints, you can recognize the control signal characteristics of the temperature signal. Using arrows and the labels (1), (2), (3) and (4), show which the graphs matches the characteristics 2 marks

1) Critically damped
2) Slightly under-damped
3) Chronically under-damped and unstable
4) Chronically over-damped



Q4C) You have passed the first interview and have been invited back after taking this quiz (I can hear the Tardis in the distance already). Briefly explain TWO key features about Fig. 1 that enable you know that you are dealing with a "microcontroller" rather than a "micro-processor . 3 marks

Lots of possible answers – **Key – many features (SPI, memory etc) part of the Blackfin Chip**, Other fetures might include Easy availability of a programming environment, lots of code examples (all over the quiz :-)

Q4D) For the Q5 interview question you asked to develop a uTTCOS task *void TardisControlTask(void)*written in C++ capable of recognizing the difference between switch presses of length 2.1 seconds, 2.5 seconds, and of course 42 seconds. Complete the following uTTCOS main( ) program that will allow you control this task 3 marks

#include <all necesssary,h> ***<MockDevices> <uTTCOS> "Lab0" – all are satisfactory (1/2 mark)***

```
                    void main(void) {
                        uTTCOS_Init( );
                        uTTCOS_InitLED( );
```

My_Init_Switches( ) ***(1/2 mark)***

```
                        uTTCOS_AddPremptive_Thread(Audio, NO_DELAY, EVERY_TIC);
                        uTTCOS_AddThread(Flash_LED5, NO_DELAY, EVERY_SECOND);
```

uTTCOS_AddThread(TardisControlTask , NO_DELAY, EVERY_SECOND / 20); // MUST BE FAST TO BE ABLE TO MEASURE 0.1 sec)

¼ mark for name ¾ mark for time

```
                        uTTCOS_Start( );
                        while (1) { uTTCOS_Sleep( );  uTTCOS_DispatchTasks( );  }
                    }
```

THERE ARE NO MARKS FOR ANSWERING THIS BONUS QUESTION FOR REALLY DEDICATED DR. WHO FANS
Comparing the amount of effort to develop an ASM program compared to a C++ program leads to the *Rule 1 "Don't write in assembly code".* What famous Dr. Who quote does this remind you of, and why?
ASM lines of code are bigger than C++ LOC – So ASM code (inside processor) is much bigger that C++ code (human read) outside the program

---

Q5) **In Q4 and Q5 you can assume you are provided with the following functions from your Lab Code -- *Flash_LED1( ), Flash_LED3( ), Flash_LED5( ), My_ReadSwitche( ), My_InitSwitches( ).*** *In this question you are asked to design and write a C++ uTTCOS task <u>void TardisControlTask(void)</u> that uses the switches to control three tasks – Flash_LED1( ), Flash_LED2( ) and TARDIS_Water_Control( ) (for this Midterm Q3).*

A) Explain why designing *<u>void TardisControlTask( )</u>* to include the following task might have an impact on the *Flash_LED5* operation – see Q4 D). **If the latest switch 4 press was held down for close to 42 seconds activate *TARDIS_Water_Control( )* to start the TARDIS coffeepot control 2 marks**
A "proper" uTTCOS does not care how long the SW4 is pressed down as it POLLS to see when SW4 changes and does not wait. However *TARDIS_Water_Control( )* function includes. If this wait is longer than 1 / 20 second, then it will upset the timing of all the other tasks in the program.

B) DESIGN and then write a C++ (NOT ASM) uTTCOS task that has the following functionality. 10 marks
B) If the latest switch 4 press was held down for close to 2.1 seconds and switch 3 is pressed – call *FlashLED1;*
C) If the latest switch 4 press was held down for close to 2.5 seconds and switch 2 is pressed – call *FlashLED3( );*
The first call to *FlashLED1* and *FlashLED3* will cause their LED's to turn on. A second call will turn their LEDs off.

C++ code (NOT ASM)	Design information
int countEntriesWhileSW4Pressed= 0; // Could be static inside function *enum {INIT, WAIT_LO, WAIT_HIGH, JUST_LOW, CONTINUE}* *#define SW4_MASK 0x8* *#define SW3_MASK 0x4* *#define SW2_MASK 0x2*	*Need to count how many entries into code* *Identify states and switches*
void <u>TardisControlTask(void) {</u>	
static int currentState = INIT; *int nextState = currentState;*	*Identify state used*
switch (currentState) { *case INIT: nextState = WAIT_LO;* *countEntriesWhileSW4Pressed = 0; break;*	*Handle init state* *PSP -- Put this everywhere in case you forget it!*
case WAIT_LOW: *countEntriesWhileSW4Pressed = 0;* *if ((My_ReadSwitches & SW4_MASK) == SW4_MASK)* *nextState = WAIT_HIGH;* *break;*	*Clear counter and move to next state if SW 4 pressed*
case WAIT_HIGH: *countEntriesWhileSW4Pressed++;* *if ((My_ReadSwitches & SW4_MASK) != SW4_MASK)* *nextState = JUST_LOW_HIGH;* *break;*	*Increment counter while SW4 is pressed and move on if released*
case JUST_LOW: *nextState = CONTINUE:* *switch (countEntriesWhileSW4Pressed) {* *case 40: case 41: Case 42: case 43:* *if My_ReadSwitches & SW3_MASK) == SW3_MASK) FlashLED1( );* *break;* *case 48: case 49: case 50: case 51:* *if My_ReadSwitches & SW2_MASK) == SW2_MASK) FlashLED3( );* *break;* *default: /* do nothing */ break;* *}* *countEntriesWhileSW4Pressed = 0;*	*Launch tasks -- Thought this was a quick solution using states* *Numbers based on entry every 1/20 second* *2.1 seconds around 40 to 43 1/20 s* ***I am giving myself a P.O.B. for the number of 'non-trivial' ways I have introduced 42 into this midterm :-)*** *2.5 seconds around 50* *PSP -- Put this everywhere in case you forget it!*
case CONTINUE: *countEntriesWhileSW4Pressed = 0;* *nextState = WAIT_LOW; break;*	*Pause state I like to code*
default: Send ErrorMessage('"Bad current state"); break; *}*	*Always have one*
currentState = nextState; *}*	*Prepare for next entry*

YOUR NAME _____ (Bonus if self-evaluation is within 10% of actual mark)

EXPECTED CLASS MARK	Q1 6 / 10.5	Q2 9 / 12	Q3 5 /11	Q4 6 / 10	Q5 9 / 12	TOTAL 35 / 55.5

RECORDED OUT OF 50 TO TAKE INTO ACCOUNT SOME PEOPLE WILL NOT FINISH ALL QUESTIONS 35 / 50 C+ / B- / B

SELF EVALUATION BONUS MARKS -- (0.5 mark each question, 2.5 marks final) -- Possible maximum mark 120%

Class Average with SE BONUS 37.5 / 50 B / B+