ENB350 Real-Time Computer-Based Systems
Laboratory Exercise 1
Introduction to Microcontroller Programming

Learning Objectives:

After completing this laboratory, you should be able to

1. use the chosen embedded systems development hardware (RCM4000 with the Rabbit4000) for application development
2. use the embedded systems development software (Dynamic C and its libraries) for application development
3. compile, run and debug programs using Dynamic C
4. modify and run a sample program that uses port read/write functions and external hardware
5. modify and run a sample program that uses the real-time clock

Reference Documentation:

The following reference book is the closest book support for the technical material.

Kamal Hyder and Bob Perrin, *Embedded System Design using the Rabbit 3000 Microprocessor – Interfacing, Networking, and Application Development*, Newnes, 2005

The reference book above is more application oriented and does not fit the requirements for a textbook in this unit. Further, note that the development boards in the laboratory are the newer Rabbit 4000 processor based, while the reference book discusses the older Rabbit 3000 based application development. There is no similar book on the newer platform. For correct details on the RCM4000 and Dynamic C you should consult user manuals and other documentation provided. These are available on Blackboard under Learning Resources – Labs and Assignment – Rabbit and Dynamic C Documents.

Another reference book *Fundamentals of Embedded Software – Where C and Assembly meet,* by Daniel W. Lewis, Prentice Hall, covers the underlying theoretical aspects better and comes with its own suite of laboratory exercises based on the Intel x86 architecture. You are not required to work out the exercises in that reference or to become familiar with another processor. This book is now out of print and has been replaced by a new version *Fundamentals of Embedded Software with the ARM Cortex-M3* which is on your reference list as a recommended reference. The ARM Cortex-M3 is specifically designed for real-time applications and 32 bit processors now account for more than 75% of new embedded applications. This book covers topics discussed in the lectures.

As an embedded real-time system developer, you could very well be faced with a situation where you have to work with a new processor and/or development platform. You will need to get hold of relevant documentation and become familiar with the relevant architectural and programming details quickly.

Information on the processor and the application development system can be found from technical documents provided by the vendor. These have been placed on the Blackboard site for the unit.
The following documents are particularly useful to start with.

- RCM4000 RabbitCore User's Manual (R4000UM.pdf)
- Rabbit 4000 Microprocessor User's Manual (RC4000UM.pdf)
- Rabbit 4000 designers handbook (R4000DH.pdf)
- Dynamic C User's Manual (DCPUM.pdf)
- Dynamic C Function Reference Manual (DynCFunRef.pdf)
- Rabbit Reference Poster (rab40_ref_poster.pdf)
- Rabbit Instruction Reference Manual (RabbitInstructions.pdf) – this is copyrighted.

Preparation activities

An introduction will be given by the tutor. Make sure that you can identify and use the following at your workstation in the laboratory.

1. The RCM4000 development boards. They are housed in boxes with a see-through hole and a clear front panel.
2. The host PC and its serial port connections. A USB-to-serial converter will be used.
3. The programming cable, its terminations and the programming port
4. The power supply input to the box. What does the RCM4000 need?
5. The network interface to the RCM4000 and network cable
6. Network (internal network and external QUT network) connections on the PC
7. Switches and LEDs on the RCM4000 board
8. Ethernet switch and connections to the board, host PC and internal network
9. Dynamic C executable on the host (you will need to log in)
10. Sample programs and Library files provided with Dynamic C and relevant to RCM4000

Get a copy of the RCM4000 Rabbit Core User's Manual and the Dynamic C user manual from Blackboard for your reference. Find out about 'costate' and 'waitfor' – these are not part of ANSI standard C. Dynamic C is more than a programming language – it is also a run-time environment facilitating multitasking.

Laboratory Tasks:

Task A. [1 Mark]
Refer to the Chapter 3 on "Running Sample Programs". Read the documentation and compile and run the program CONTROLLED.C in the SAMPLES/RCM4000 folder.

- Which parallel port and which pins are the LEDs mapped to?
- Note that once your program is compiled on the PC and loaded on the RCM4000 board, it will run on the board without a need to connect to the host. However, the monitor program maintains a STDIO window and uses it as a serial terminal to which the embedded program can send output or received input from. Find the function used to display the output and examine the characters the precede the actual string to be displayed. What is the need for these additional characters?

Task B. [1 Mark]
Compile and run the program TOGGLESWITCH.C in RCM4000 sample program folder.

- What is debouncing of a switch? How is it achieved?

Task C. [3 Marks]
A real-time clock is a free running oscillator independent of the system clock that drives the central processing unit. It is available as an internal or external peripheral and often includes counters and registers to keep track of the time elapsed since the system was last initialized (a reference time) and the time-of-day in years, months, days, hours, minutes and seconds. Library functions are usually available to user programs to read these and make use of the real time value. The real-time clock can also generate periodic interrupts to the central processing unit which can trigger the time update operations in software and are used by a task scheduler to keep track of time ticks or slices and decide when to switch from one task to another in a multitasking environment.

Find the program RTC_TEST.C in RTCLOCK sample program folder, which demonstrates the use of real-time clock functions. Compile and run the program. Modify this program such that it uses the RTC functions to
(a) set the current time to a reference value when the program starts and
(b) periodically updates this time say every 10 seconds and displays it on the console and
(c) remains in the background allowing another task to run by using costates. Go to the sample program FlashLED1.c. Cut and paste and modify code such that LED1 will flash remaining on for 1

second and off for 1 second while the real-time clock is running and the time is updated every 10 seconds and displayed on the console.

```
#class auto

char bfr[25];
unsigned long     t1, t2;

// function prototype for print_time function
void print_time(unsigned long);
// include the definition of this function from the sample code

void DispStr(int x, int y, char *s)
{
   x += 0x20;
   y += 0x20;
   printf ("\x1B=%c%c%s", x, y, s);
}


int main()
{
      auto unsigned int j;
      auto struct tm    rtc;

 // change the date/time via tm_wr to desired reference
 // include code for this from the sample code

   while(1)
   {
    costate
    {
     waitfor(DelaySec(10L));
      t2 = read_rtc();        // read the RTC
     print_time(t2);          // Display updated time
    }

    costate
    {
     //other FSM
     // include code to turn an LED ON and OFF with 1 sec
     // delay in between each change
     yield;
    }

   }

      return(0);
}
```

Note the two costate statements in the above program. These are not part of ANSI standard C. Each will continue to execute and could potentially be in an infinite loop. When does program control go from one costate to another? Although the syntax of Dynamic C and the architecture of R4000 are particular cases of system development, concepts such as real-time clock and multitasking are part of all real-time embedded application development.