Bournemouth University

School of Design, Engineering and Computing

Course:	**B.Sc. Computing**
	B.Sc. Software Engineering Management
Unit:	**Systems Architecture**
Assignment:	**Two**
Due Date:	**7th March 2005**

Aims and Objectives

This assignment provides the student with the opportunity to demonstrate their ability to work on thair own, to practice their ability to view their own work in a critical light, in the process of demonstrating their understanding of low level programming.

The assignment is intended to address the following learning outcomes:

1: Describe the format of simple data structures and their limitations.

3: Understand the relationship between computer hardware and software.

4: Relate basic constructs of high-level programming language to their low level implementation.

5: Explain how a computer system processes real-time events.

The Task

You are to work independently to provide the software compenent for a more complex alarm clock than that given in the first assignment. **You are to complete the `tick` and `keyboard` subroutines outlined in the *skeleton* file (`clock.s`).** A test harness (`harness.s`) is also provided to assist with testing, but this should not be modified.

The `tick` subroutine is called once every second; it is responsible for updating the current time and checking it against the alarm setting. The alarm time (`Alarm`) and current time (`Time`) are stored as hours and minutes in BCD form. Setting the single byte variable `Klaxon` to any non-zero value will turn on the alarm, while setting the variable to zero will turn it off.

When the current time is the same as the alarm time, the klaxon should be sounded. The klaxon should be allowed to continue sounding for an hour after the alarm time. If the user presses the snooze key the klaxon should be silenced for up to ten minutes. *You do not need to implement the snooze feature.*

The `Keyboard` subroutine is called when a key on the control panel is pressed. It should read the keyboard matrix from the single-byte variable `KeyCode` with the following values:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Unused	Unused	Unused	Snooze	Minute	Hour	Alarm	Time

The keys are pressed in a combination. The user presses the Time key and either the Hour or Minute keys. The `keyboard` routine will increment either the hour count or the minute count accordingly. Similarly the Alarm key and the Hour or Minute keys will increment either the hour or minute count for the alarm setting.

A test harness is provided to allow you to test your code. To run the software you should:

1. Create a new project, (called `clock.apj`); you should add the file `clock.s` to the new project. As this includes the test harness you **should not** add `harness.s` to the project.
2. You should now build the project to test out the skeleton and test harness.
3. Now run the debugger to get a feel for the test harness and debug environment.
4. To run the test harness you must first "step" over the initialisation code added by the assembler.
5. You can now "step into" the `Main` program.
6. You should now "step", which will call the test harness. This will report the current time and alarm setting in the Console window before asking for your input. If you use the "step into" function you will have a large number of steps to make before the system stops to await your input.
7. When the test harness is ready to call your code (`keyboard` or `tick`) it will return from the test harness and allow you to step through, and thus debug, your code.

The test harness supports the following operations:

`T`	Time	Pressing the `T` key will place the harness into "Time Mode". This is as if the user has pressed the Time key. Pressing either the `H` or `M` keys will call the `keyboard` routine with the *Time* bit set and either the *Hour* or *Minute* bit set according to which key was pressed.
`A`	Alarm	Places the harness into "Alarm Mode". This is as if the user has pressed the Alarm key. Pressing either the `H` or `M` keys will call the `keyboard` routine with the *Alarm* bit set and either the *Hour* or *Minute* bit set depending on which key was pressed.
`H`	Hour	Set the *hour* bit in the keyboard matrix and call the `keyboard` subroutine. If a mode has been selected, the mode bits will be set accordingly (*Time* or *Alarm*), however, if a mode has not been selected neither of the mode bits will be set, and the `keyboard` routine should ignore the request.
`M`	Minute	Set the *minute* bit in the keyboard matrix and call the `keyboard` subroutine. If a mode has been selected, the mode bits will be set accordingly (*Time* or *Alarm*), however, if a mode has not been selected then neither of the mode bits will be set and the `keyboard` routine should ignore such a request.
⟨*Space*⟩	Tick	This will call the `tick` subroutine. As the `tick` routine is to count seconds rather than minutes you should press the space bar 60 times for each minute.
`S`	Snooze	Set the *snooze* bit in the keyboard matrix and calls the `keyboard` subroutine. The klaxon should be silenced (turned off) for a short period of up to 10 minutes. *This is an optional feature and you are not required to implement this.*
`Q`	Quit	Will exit the test harness are return to the monitor.

The following notes may help:

1. The `Time` and `Alarm` values are stored as 32-bit words in Binary Coded Decimal format.
2. The hours and minutes are stored as two 8-bit values in the lower half of a 32-bit word. The highest value for the lower byte (minutes) is 59. Hence you must reset the value to zero and increment the upper byte (hours) in your code.

3. Remember to use "step" to step over of call to the test harness, this is the instruction:

   ```
   BLAL   Harness
   ```

   On line 11 of the `clock.s` skeleton file. This will make single stepping though your code much easer as you will be able to ignore the test harness.

Deliverables

You are expected to hand in a single coursework report containing the following:

- A Coursework Report sheet completed with your name and lab group.
- A written description of the operation of your `tick` and `keyboard` subroutines.
- Design documentation for the two subroutines, including any pseudo-code, flow-charts, test programs, or any other documentation you may have produced.
- A print out of fully commented program listing.
- An assessment of your own work. You should give yourself a mark out of 10, where 0 is considered to be very poor and 10 is outstanding. You must use one or two paragraphs to justify your self-assessment. You should include a couple of paragraphs on what you have learned by attempting this assignment.
- A critical review of the assignment. This should be no more than two or three paragraphs.

You will be required to demonstrate your solution in the first seminar after the hand-in date. It is expected that this demonstration should last for no more than five minutes. The demonstrations should be carried out in an informal, but professional manner.

Marking Scheme

Description		Design Documentstion		Commented Code		Persional Development	
`tick`	10	`tick`	10	`tick`	10	Self Assessment	10
`keyboard`	10	`keyboard`	10	`keyboard`	10	Critical Review	10
				Snooze	10	Demonstration	10

Signatures

LECTURER ....

QUALITY ASSESSOR ....

clock.s — skeleton Clock file for you to complete

```
        TTL     – Alarm Clock – <Put Your Name Here>

; ============================================================
        AREA  Program, CODE, READONLY
; ============================================================

; Make things just a bit easier to debug !

        ENTRY
Main
        BLAL  Harness          ; Call the test harness
        LDR   LR, =Main        ; Set Return address
        MOV   PC, R12          ; Call Students Code

; ============================================================
; Include the Test Harness
; ============================================================

        INCLUDE harness.s
        OPT     1          ; Turn Listing output on

; ============================================================
        AREA  Assignment, CODE, READONLY
; ============================================================

; ============================================================
; Tick
; ============================================================
;
; This routine is called once every second. It should:
;
;   Increment a seconds counter
;   If seconds counter == 60 then
;       reset seconds counter to zero
;       increment current time
;          increment minute count
;          if minute count == 60 then
;              reset minute count to zero
;              increment hour count
;              if hour count == 13 then
;                  reset hour count to one
;              end if
;          end if
;       Ensure Klaxon is off
;       if current time == alarm time then
;           Turn Klaxon On
;       end if
;   end if

Tick
        NOP                ; Your code goes in here.
```

```
        MOV   PC,LR ; Return to harness

        OPT    4          ; New page

; ==========================================================
; Keyboard
; ==========================================================
;
; This routine is called when the user presses any of the control buttons on the alarm
; clock. The keyboard consists of just four keys. The KEYCODE variable holds the value
; for the keyboard matrix as follows:
;
; Bit 7   Bit 6   Bit 5   Bit 4   Bit 3   Bit 2   Bit 1   Bit 0
; Unused  Unused  Unused  Snooze  Minute  Hour    Alarm   Time
;
; To set the alarm time the user must hold the "Alarm" key down and press the "Hour"
; key to increment the Alarm time by one hour.
;
; This routine should read the keyboard by reading the KEYCODE variable and modify
; the Current Time or the Alarm time accordingly.

Keyboard
        LDRB R0, KeyCode
        NOP               ; Your code goes in here.

        MOV   PC, LR ; Return to harness


        OPT    4          ; New Page

; ==========================================================
        AREA  Data, DATA
; ==========================================================

; Current time and Alarm time are both stored as Binary Coded Decimal values.
; I.e., $1234 represents 12:34 or thirty four minutes past 12.

Time    DCD    0x1234 ; Current Time in BCD (12:34)
Alarm   DCD    0x1240 ; Alarm Time in BCD  (12:40)

Klaxon  DCB    0      ; $00 is Off, $FF is On
KeyCode DCB    0      ; Keyboard Matrix

; Any additional variable you require should be placed here.

        END
```

```
        OPT    2                ; Turn Listing Off

; =====================================================
; Test Harness - DO NOT CHANGE
; =====================================================

        AREA   TestHarness, CODE

Harness
        MOV    R11, LR          ; Save the return address

        LDR    R0, HReturn
        ORRS   R0, R0, R0
        MOVNE  PC, R0

; Report current status to the user
HStatus
        LDR    R10, =MSG        ; Display Time
        BLAL   ASCIIZ
        LDR    R0, Time         ; Read time
        BLAL   ShowTime         ; Display it

        BLAL   ASCIIZ           ; Display Alarm setting
        LDR    R0, Alarm        ; Read Alarm setting
        BLAL   ShowTime         ; Display it

        LDRB   R0, Klaxon       ; Read Klaxon Status

        ORRS   R0, R0, R0       ; Is it clear ?
        BLNE   ASCIIZ           ; No => Display Status

; Prompt the user for the next command

        LDR    R10, =Prompt     ; Point to prompt text
        BLAL   ASCIIZ

        SWI    &4               ; Read char from keyboard

        CMP    R0, #'a'         ; Is char lower case?
        SUBGE  R0, R0, #' '     ; Yes => Convert to upper case


; Lookup command in command table

        LDR    R10, =Command    ; Start of command table
        EOR    R3, R3, R3       ; Counter (command)

HCmd    LDRB   R1, [R10], #1    ; Read command letter
        ORRS   R1, R1, R1       ; Is it end of table?
        BEQ    HStatus          ; No => Go around again
```

```
        CMP   R1, R0            ; Is it this command?
        BEQ   doCommand         ; Yes => Execute it

        ; Skip to next command letter
HSkip   LDRB  R1, [R10], #1
        ORRS  R1, R1, R1        ; Is it zero?
        BNE   HSkip
        ADD   R3, R3, #1        ; Increment command count
        BAL   HCmd              ; Look at next command

doCommand
        ; Display command text
        BLAL  ASCIIZ

        LDR   R10, =JumpTable ; Point to jump table

        LDR   PC, [R10, R3, LSL #2] ; Jump to function

; ======================================================
;
; Time Mode (T)
; Modify current time
; Set the Time bit (bit 0) in the Keyboard matrix.

HTime
        MOV   R0, #0x01         ; 0000 0001 (Time Bit)
        STRB  R0, KeyCode       ; Set Keyboard Matrix
        BAL   HStatus

; ======================================================
;
; Alarm Mode (A)
; Modify Alarm time
; Set the Alarm bit (bit 1) in the Keyboard matrix.

HAlarm
        MOV   R0, #0x02         ; 0000 0010 (Alarm Bit)
        STRB  R0, KeyCode       ; Set Keyboard Matrix
        BAL   HStatus

; ======================================================
;
; Snooze (S)
; Hit the Snooze button - Set the Snooze bit (bit 4)
; in the Keyboard matrix and then call the students
; keyboard handler.

HSnooze
        MOV   R0, #0x10         ; 0001 0000 (Snooze Bit)
        BLAL  HKeyboard         ; Call Student's code

        EOR   R0, R0, R0
        STRB  R0, KeyCode       ; Clear Keyboard Matrix
        BAL   HStatus
```

```
; ==================================================
;
; Hours (H)
; Increment Hours of current mode (Time or Alarm)
; Set the Hour bit in the keyboard matrix, leaving
; the current mode set. Then call the students keyboard
; handler.

HHour
        MOV  R1, #0x04        ; 0000 0100 (Hour Bit)
        BAL  HKey             ; Call Student's code

; ==================================================
;
; Minutes (M)
; Increment Minute of current mode (Time or Alarm)
; Set the Minute bit in the keyboard matrix, leaving
; the current mode set. Then call the students keyboard
; handler.

HMinute
        MOV  R1, #0x08        ; 0000 1000 (Minute Bit)

; This code can lead to the user pushing the Minute or
; Hour button without setting a Mode. The students code
; is going to have to process this  possibility.

HKey
        LDRB R0, KeyCode      ; Read current matrix

        AND  R0, R0, #0x03    ; Keep the Mode bits
        ORR  R0, R0, R1       ; Set Key bit (H or M)
        BLAL HKeyboard        ; Call Student's code
        BAL  HStatus

HKeyboard
        MVN  R1, R0
        ORR  R0, R0, R1, LSL #5
        STRB R0, KeyCode

        STR  LR, HReturn      ; Save return address

        LDR  R12, =UserCode
        LDR  R12, [R12]
        MOV  PC, R11          ; Call Student's code

; ==================================================
;
; Tick (Space)
; This is a bit of a cheat. It is here to allow the student
; to test their tick code without worrying about it ticking
; away every second.
```

```
HTick
        EOR   R0, R0, R0
        STR   R0, HReturn      ; Save Harness Status

        LDR   R12, =UserCode
        LDR   R12, [R12, #4]
        MOV   PC, R11          ; Call Student's code

; ==================================================
;
; Quit (Q)
; Stop the harness and return to the monitor

HQuit
        SWI   &11

; ==================================================
; Display - Display the time in R0

ShowTime
        MOV   R9, LR           ; Save Return address

        AND   R1, R0, #0xFF ; R1 = Minutes
        MOV   R2, R0, LSR #8 ; R2 = Hours

        MOV   R0, R2
        BLAL  ShowTwo          ; Display Hours

        MOV   R0, #':'
        SWI   &0               ; Display ":"

        MOV   R0, R1           ; Display Minutes
        MOV   LR, R9           ; Recover Return Address

ShowTwo
        MOV   R3, R0           ; Save Second nibble
        MOV   R0, #'0'         ; ASCII "0"
        ADD   R0, R0, R3, LSR #4 ; Add first nibble
        SWI   &0               ; Display it

        MOV   R0, #'0'         ; ASCII "0"
        AND   R3, R3, #0xF     ; Disregard first Nibble
        ADD   R0, R0, R3       ; Add to ASCII "0"
        SWI   &0               ; Display char

        MOV   PC, LR           ; Return

; Display ASCIIZ String
; R10 points to the zero terminated string
; R0 reset to zero

ASCIIZ LDRB R0, [R10], #1      ; Read in the character, inc R10
        CMP   R0, #0           ; Is char zero terminator ?
        MOVEQ PC, LR           ; Yes => Return to caller
```

```
        SWI     &0              ; No => Display char (R0)
        BAL     ASCIIZ          ; Next char

        OPT     4               ; New Page

; ==========================================================
; Private Data Section for the Test Harness
; ==========================================================

;       AREA    HarnessData, DATA

MSG     DCB     13,10,"Time: ",0
        DCB     " Alarm: ",0
        DCB     " [Alarm]",0

Prompt  DCB     " ? ",0

Command
        DCB     "T", 13,10,"Time Mode",0
        DCB     "A", 13,10,"Alarm Mode",0
        DCB     "H", "our",0
        DCB     "M", "inute",0
        DCB     "S", "nooze",0
        DCB     " ", "Tick",0
        DCB     "Q", "uit",13,10,0
        DCB     0

        ALIGN
JumpTable
        DCD     HTime
        DCD     HAlarm
        DCD     HHour
        DCD     HMinute
        DCD     HSnooze
        DCD     HTick
        DCD     HQuit

UserCode
        DCD     Keyboard
        DCD     Tick

HReturn DCD     0

; ==========================================================
; End of Test Harness
; ==========================================================

        OPT     4               ; New Page

        END
```