Capstone Design Course

Lecture-7: Programs

By

Syed Masud Mahmud, Ph.D.

Copyright © 2002 by Syed Masud Mahmud
A Program using OC3 Interrupt

- We want to write a program to control LED1 and LED2 as follows:
  - Flashing of LED2 will be controlled using OC3 service routine.

- Flashing of LED2 will be controlled using OC3 service routine.

<table>
<thead>
<tr>
<th>Sw</th>
<th>LED1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>OFF</td>
</tr>
<tr>
<td>Closed</td>
<td>ON</td>
</tr>
</tbody>
</table>

LED2 is to be flashed once per second using OC3 function.
A Program using OC3 Interrupt (contd.)

- Our program will have two parts: the MAIN Program and the OC3 Service Routine.
- The MAIN program will be used to control LED1 using the switch Sw.
- The OC3 service routine will flash LED2 once per second.
- In order to flash LED2 once per second, we need to keep it ON for 0.5 sec and then OFF for 0.5 sec, and we should repeat this ON and OFF sequence for ever.
A Program using OC3 Interrupt (contd.)

• We will set up OC3 function in such a way that the processor will be interrupted once every 20 msec by the OC3 function.
• Thus, the processor will enter into the OC3 Service Routine once every 20 msec.
• The OC3 service routine will keep track, how many times the processor is entering the routine.
• Since 20 msec x 25 = 500 msec = 0.5 sec, the OC3 service routine will toggle LED2 after every 25 interrupts.
* Define Registers and Parameters *

REGBAS EQU $1000
PORTA EQU 0
TCNT EQU $0E
TOC3 EQU $1A
TMSK1 EQU $22
TFLG1 EQU $23
TWENTY_MS EQU 40000
HALF_SEC EQU 25

* 
ORG $200

OC3_CNT RMB 1

* OC3 VECTOR *

ORG $00D9 ; OC3 Vector for running
JMP OC3SERV ; from RAM
MAIN PROGRAM STARTS FROM HERE

ORG $1040 ; start code at $1040

MAIN
CLR OC3_CNT
LDX #REGBas
LDD TCNT,X
ADDD #TWENTY_MS
STD TOC3,X
BSET TMSK1,X $20
BCLR TFLG1,X $DF
CLI

LOOP
BRSET 0,X 1 ON
BCLR 0,X $20
BRA LOOP

ON BSET 0,X $20
BRA LOOP
OC3SERV LDD TOC3,X ; Read TOC3 and then
ADDD #TWENTY_MS ; add value for another
STD TOC3,X ; 20ms delay
INC OC3_CNT
LDAA #HALF_SEC
CMPA OC3_CNT
BNE OC3DONE
LDAA PORTA,X
EORA #$40 ; Toggle Line PA6
STAA PORTA,X ; to flash LED2
CLR OC3_CNT
OC3DONE BCLR TFLG1,X $DF ; Clear OC3F bit
RTI
END
This program uses OC2 interrupt to flash lines PD2, PD3, PD4 and PD5 as follows:

<table>
<thead>
<tr>
<th>Line</th>
<th>Flashing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD2</td>
<td>50 times per second</td>
</tr>
<tr>
<td>PD3</td>
<td>25 times per second</td>
</tr>
<tr>
<td>PD4</td>
<td>Once per second</td>
</tr>
<tr>
<td>PD5</td>
<td>Once every two seconds</td>
</tr>
</tbody>
</table>

REGBAS EQU $1000
TCNT EQU $0E
TOC2 EQU $18
TMSK1 EQU $22
TFLG1 EQU $23
TMSK2 EQU $24
TFLG2 EQU $25
PORTD EQU 8
DDRD EQU 9
TEN_MS EQU 20000
TWENTY_MS EQU 2
HALF_SEC EQU 25
ONE_SEC EQU 2
ONE_MIN EQU 60
ORG $200
OC2_CNT RMB 1
TWENTYMS_CNT RMB 1
HALFSEC_CNT RMB 1
SEC_CNT RMB 1
OC2 Prog.asm (contd.)

*---------------------------------------------------------------
* OC2 VECTOR
* Remove the following two lines or put a comment character `*` at the
* beginning of each line if you want to run the program from an external
* EEPROM.
*---------------------------------------------------------------

ORG $00DC          * OC2 Vector for running
JMP OC2SERV        * from RAM

*---------------------------------------------------------------
* If you want to run this program from the external EEPROM then remove
* the comment character `*` from the beginning of the following seven lines.
*---------------------------------------------------------------

ORG $FFE6          * OC2 Vector for running
FDB OC2SERV        * from external EEPROM

ORG $0000          * Start code at $E000
START               
LDS #$23FF          * Initial value of stack pointer
* Remove the line ORG $1040 or put a comment character '/*' at the beginning of this line if you want to run the program from an external EEPROM.

    ORG    $1040    * start code at $1040

*-------- MAIN PROGRAM STARTS FROM HERE *--------

MAIN

    CLR    OC2_CNT
    CLR    TWENTYMS_CNT
    CLR    HALFSEC_CNT
    CLR    SEC_CNT
    LDX    #$REGBA
    LDAA    #$FF
    STAA    DDRD,X
    LDD    TCNT,X ; Read timer
    ADDD    #$TEN_MS ; Add value for 10ms
    STD    TOC2,X ; delay and store in TOC2
    BSET    TMSK1,X $40 ; Set OC2I bit
    BCLR    TFLG1,X $BF ; Clear OC2F bit

LOOP

    NOP
    BRA    LOOP
SERVICE ROUTINE FOR OUTPUT COM Pare 2

OC2SERV

LDD TOC2,X ; Read TOC2 and then
ADDD #TEN_MS ; add value for another
STD TOC2,X ; 10ms delay

GO TO 10ms ROUTINE TO DO WHATEVER YOU ARE SUPPOSED
TO DO EVERY 10ms.

JSR TENMS RTN
INC OC2_CNT
LDAA #TWENTY_MS
CMPA OC2_CNT
BEQ TWENTYMS_TASK
JMP OC2DONE

GO TO 20ms ROUTINE TO DO WHATEVER YOU ARE SUPPOSED
TO DO EVERY 20ms.
OC2Prog.asm (contd.)

TWENTYMS_TASK   JSR      TWENTYMS_RTN
    CLR      OC2_CNT
    INC      TWENTYMS_CNT
    LDA A    #HALF_SEC
    CMP A    TWENTYMS_CNT
    BEQ      HALFSEC_TASK
    JMP      OC2DONE

*  
*  GO TO 0.5sec ROUTINE TO DO WHATEVER YOU ARE SUPPOSED
* TO DO EVERY 0.5sec.
*  
HALFSEC_TASK    JSR      HALFSEC_RTN
    CLR      TWENTYMS_CNT
    INC      HALFSEC_CNT
    LDA A    #ONE_SEC
    CMP A    HALFSEC_CNT
    BEQ      SEC_TASK
    JMP      OC2DONE

*  
*  GO TO 1sec ROUTINE TO DO WHATEVER YOU ARE SUPPOSED
* TO DO EVERY SECOND.
SEC_TASK                JSR    ONESEC_RTN
CLR          HALFSEC_CNT
INC          SEC_CNT
LDA A         #ONE_MIN
CMPA          SEC_CNT
BEQ           MIN_TASK
JMP           OC2DONE

*                 
*     GO TO 1min ROUTINE TO DO WHATEVER YOU ARE SUPPOSED 
* TO DO EVERY MINUTE. 
*                 
MIN_TASK                JSR    ONEMIN_RTN
CLR          SEC_CNT
OC2DONE BCLR  TFLG1,X $BF     ; Clear OC2F bit
RTI

*                 
*     END OF SERVICE ROUTINE FOR OUTPUT COMPARE 2 
*
OC2Prog.asm (contd.)

*---------------------------------
* THIS ROUTINE IS EXECUTED ONCE EVERY 10ms
*---------------------------------

* TENMS_RTN
PSHA
LDAA PORTD,X
EORA #4 ; Line PD2 flashes
STAA PORTD,X ; 50 times a second
PULA
RTS

*---------------------------------
* THIS ROUTINE IS EXECUTED ONCE EVERY 20ms
*---------------------------------

* TWENTYMS_RTN
PSHA
LDAA PORTD,X
EORA #8 ; Line PD3 flashes
STAA PORTD,X ; 25 times a second
PULA
RTS
* THIS ROUTINE IS EXECUTED ONCE EVERY HALF SECOND

HALFSEC_RTN
    PSHA
    LDAA PORTD,X
    EORA #16 ; Line PD4 flashes
    STAA PORTD,X ; once per second
    PULA
    RTS

* THIS ROUTINE IS EXECUTED ONCE PER SEC

ONESEC_RTN
    PSHA
    LDAA PORTD,X
    EORA #32 ; Line PD5 flashes
    STAA PORTD,X ; Once every two seconds
    PULA
    RTS

* THIS ROUTINE IS EXECUTED ONCE PER MIN

ONEMIN_RTN
    PSHA
    PULA
    RTS
    END
A Program to display some messages and numbers on the LCD: DISPLAY.asm

*******************************************************************************
* This program displays some messages and numbers on the LCD
*******************************************************************************

LCDBAS EQU $B5F0         * LCD port address

* Variables of the program
*

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>$200</td>
<td></td>
</tr>
<tr>
<td>CHAR1</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>CHAR2</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>CHAR3</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>HOUR</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>MIN</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>SEC</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>AMPM</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>RMB</td>
<td>1</td>
</tr>
<tr>
<td>DISTANCE</td>
<td>RMB</td>
<td>1</td>
</tr>
</tbody>
</table>

* Locations for keeping three ASCII codes
* location for HOUR
* location for MIN
* location for SEC
* location for AMPM 0:AM 1:PM
* location for WEIGHT
* location for DISTANCE
* If you want to run this program from the external EEPROM then
* remove the comment character '* ' from the beginning of the
* following five lines.

* ORG $FFE
* FDB START * Reset vector
* ORG $E000 * Start code at $E000
*START NOP
* LDS #$23FF * Initial value of stack pointer

* Remove the line ORG $1040 or put a comment character ' * ' at the
* beginning of this line if you want to run the program from an
* external EEPROM.

ORG $1040 * start code at $1040

JMP MAIN * jump over data to start of program
<table>
<thead>
<tr>
<th>MESS1</th>
<th>FCC</th>
<th>'ECE4600 WINTER-2001'</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESS2</td>
<td>FCC</td>
<td>'TIME :: '</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
<tr>
<td>MESS3</td>
<td>FCC</td>
<td>'Weight = '</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
<tr>
<td>MESS4</td>
<td>FCC</td>
<td>'lbs'</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
<tr>
<td>MESS5</td>
<td>FCC</td>
<td>'Distance = '</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
<tr>
<td>MESS6</td>
<td>FCC</td>
<td>'ft'</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
<tr>
<td>AM</td>
<td>FCC</td>
<td>'AM'</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
<tr>
<td>PM</td>
<td>FCC</td>
<td>'PM'</td>
</tr>
<tr>
<td></td>
<td>FCB</td>
<td>0</td>
</tr>
</tbody>
</table>
DISPLAY.asm (contd.)

* Main program starts from here
*
MAIN    SEI           * Disable interrupts
       JSR   LCDSET      * Setup LCD

* Load locations HOUR, MIN, SEC, WEIGHT and DISTANCE with some
* arbitrary numbers just for testing purpose.
*
   LDAA  #5
   STAA  HOUR
   LDAA  #46
   STAA  MIN
   LDAA  #7
   STAA  SEC
   LDAA  #0
   STAA  AMPM         * Time is 5:46:07 AM
   LDAA  #156
   STAA  WEIGHT       * Weight is 156 lbs.
   LDAA  #78
   STAA  DISTANCE     * Distance is 78 ft.
* Display different items on the LCD *

LDX #MESS1
JSR PRINT  * Display 'ECE4600 WINTER-2001'

JSR ROW2  * Move the cursor to Row-2
LDX #MESS2
JSR PRINT  * Display 'TIME :: '
JSR DSPTIME * Display TIME on the LCD

JSR ROW3  * Move the cursor to Row-3
LDX #MESS3
JSR PRINT  * Display 'Weight = '
LDAA WEIGHT
JSR DSPBYTE3 * Display the value of weight
LDX #MESS4
JSR PRINT  * Display ' lbs'

JSR ROW4  * Move the cursor to Row-4
LDX #MESS5
JSR PRINT  * Display 'Distance = '
LDAA DISTANCE
JSR DSPBYTE3 * Display the value of distance
LDX #MESS6
JSR PRINT  * Display ' ft'
BRA *
* This routine reads time from locations HOUR, MIN, SEC, and
* AM/Pm, and then displays the TIME on the LCD
*
DSPTIME PSHA
LDAA HOUR
Jsr DSPBYTE4 * Display HOUR
LDAA :
Jsr LCDOUT * Display :
LDAA MIN
Jsr DSPBYTE5 * Display MIN
LDAA :
Jsr LCDOUT * Display :
LDAA SEC
Jsr DSPBYTE5 * Display SEC
LDAA AM/Pm
Beq DSPAM
Ldx #PM
Jsr PRINT * Display ' PM'
Bra DONTIME
DSPA Ldx #AM
Jsr PRINT * Display ' AM'
DONTIME Pula
Rts
This routine converts an 8-bit binary number into three ASCII codes. The 8-bit binary number must be passed to this routine using Reg. A

INPUT: Reg. A   OUTPUT: CHAR1, CHAR2, CHAR3

BIN2CHR    PSHA    * save registers on the stack
PSHB
TAB       * transfer A to B
PSHX
CLRA   * A = 0
LDX #10 *
IDIV        * divide register D by 10
ADDB #$30 *
STAB CHAR3 * store ASCII code of the least sig. digit
PSHX *
PULA     * move quotient
PULB * to register D
LDX #10 *
IDIV        * divide register D by 10
ADDB #$30 *
STAB CHAR2 * store ASCII code of the middle digit
PSHX       * move quotient
PULA       * from register X
PULB       * to register D
ADDB #$30
STAB CHAR1 * store ASCII code of the most sig. digit
PULX
PULB
PULB
PULA       * get back registers
RTS
* This routine displays the value of an 8-bit number on the LCD
* in decimal. The 8-bit number must be passed to this routine
* through register A LEADING ZEROS OF THE NUMBER ARE NOT DISPLAYED
*
* INPUT: Reg. A  OUTPUT: LCD Display
*----------------------------------------------------------

DSPBYTE1 PSHA

    JSR    BIN2CHR
    LDAA   CHAR1
    CMPA   #$30
    BEQ    DSPBY11

    JSR    LCDOUT  * display most significant digit

DSPBY11 LDAA CHAR2

    CMPA   #$30
    BEQ    DSPBY12

    JSR    LCDOUT  * if CHAR2 is a zero then go to DISPCH3

DSPBY12 LDAA CHAR3

    JSR    LCDOUT  * display the next digit on the LCD
    PULA
    RTS

    * display the least significant digit
    * get back registers
* This routine displays the value of an 8-bit number on the LCD in decimal. The 8-bit number must be passed to this routine through register A
* LEADING ZEROS OF THE NUMBER ARE ALSO DISPLAYED
*
* INPUT: Reg. A OUTPUT: LCD Display
*
DSPBYTE2 PSHA * save registers on the stack
  JSR BIN2CHR
  LDA A CHAR1
  JSR LCDOUT * display most significant digit
  LDA A CHAR2
  JSR LCDOUT * display the next digit on the LCD
  LDA A CHAR3
  JSR LCDOUT * display the least significant digit
  PULA PULL
  RTS * get back registers
DISPLAY.asm (contd.)

* This routine displays the value of an 8-bit number on the LCD in
  * decimal. The 8-bit number must be passed to this routine through
  * register A. Three character spaces will be used on the LCD unit,
  * but
  * LEADING ZEROS OF THE NUMBER WILL BE REPLACED BY SPACES
  *
  * INPUT: Reg. A                               OUTPUT: LCD Display
  *
DSPBYTE3  PSHA  * save registers on the stack
  JSR        BIN2CHR
  LDAA       CHAR1
  CMPA       #$30
  BNE        DSPBY31
  LDAA       #$20
DSPBY31  JSR        LCDOUT  * display most significant digit
  LDAA       CHAR2
  CMPA       #$30
  BNE        DSPBY32
  LDAA       #$20
DSPBY32  JSR        LCDOUT  * display the next digit on the LCD
  LDAA       CHAR3
  JSR        LCDOUT  * display the least significant
  PULA
  RTS  * get back registers
* This routine displays the value of an 8-bit number (0 - 99) on the LCD in decimal. The 8-bit number must be passed to this routine through register A.
* LEADING ZERO IS REPLACED BY A SPACE CHARACTER

**INPUT:** Reg. A (range: 0 - 99)  
**OUTPUT:** LCD Display

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPBYTE4 PSHA</td>
<td>* save registers on the stack</td>
</tr>
<tr>
<td>JSR BIN2CHR</td>
<td></td>
</tr>
<tr>
<td>LDAA CHAR2</td>
<td></td>
</tr>
<tr>
<td>CMPA #$30</td>
<td></td>
</tr>
<tr>
<td>BNE DSPBY41</td>
<td></td>
</tr>
<tr>
<td>LDAA #$20</td>
<td></td>
</tr>
<tr>
<td>DSPBY41 JSR LCDOUT</td>
<td>* display most significant digit</td>
</tr>
<tr>
<td>LDAA CHAR3</td>
<td></td>
</tr>
<tr>
<td>JSR LCDOUT</td>
<td>* display the least significant digit</td>
</tr>
<tr>
<td>PULA</td>
<td>* get back registers</td>
</tr>
<tr>
<td>RTS</td>
<td></td>
</tr>
</tbody>
</table>
* This routine displays the value of an 8-bit number (0 - 99) on the
* LCD in decimal. The 8-bit number must be passed to this routine
* through register A
* LEADING ZERO IS ALSO DISPLAYED
*
* INPUT: Reg. A  (range: 0 - 99)                        OUTPUT : LCD Display
*
DSPBYTE5 PSHA
  JSR      BIN2CHR
  LDA A    CHAR2
  JSR      LCDOUT
  LDA A    CHAR3
  JSR      LCDOUT
  PULA
  RTS

* save registers on the stack
* display most significant digit
* display the least significant digit
* get back registers
* This routine sends a message to the LCD. Register X must point to * the message before calling this routine.

* PRINT      PSHA
PR01        LDA A  0,X  * Get a character from the message
   BEQ      PRDONE  * If end of message then done
   JSR      LCDOUT  * Else send the character to the LCD
   INX
   BRA      PR01

PRDONE      PULA
   RTS

* This routine moves the cursor to ROW-1

* ROW1      PSHA
   LDA A  #$02
   STA A  LCDBAS
   JSR    DELAY1
   PULA
   RTS
* This routine moves the cursor to ROW-2
*
ROW2
  PSHA
  LDAA  #$a8
  STAA  LCDBAS
  JSR   DELAY1
  PULA
  RTS

* This routine moves the cursor to ROW-3
* 
ROW3
  PSHA
  LDAA  #$94
  STAA  LCDBAS
  JSR   DELAY1
  PULA
  RTS

* This routine moves the cursor to ROW-4
*
ROW4
  PSHA
  LDAA  #$d4
  STAA  LCDBAS
  JSR   DELAY1
  PULA
  RTS
**DISPLAY.asm (contd.)**

* This routine sends a character to the LCD
* This version adjusts for 4x20 display if you're expecting
  * continuous rows 1,2,3,4

**LCDOUt**

```assembly
ldab        LCDBAS       * get current address
CMPB        #$13         * at end of first row
beq          lcd_ad1     * jif yes
CMPB        #$53         * at end of 2nd row
beq          lcd_ad2     * jif yes
CMPB        #$27         * at end of 3rd row
beq          lcd_ad3     * jif yes
STAA        LCDBAS+1     * write data to lcd output port
JSR          DELAY1
RTS
```