Lecture 12: PI/T parallel I/O, part II

- Terms and definitions
- PI/T modes of operation
  - Modes and sub-modes
- An example in C language
Term and symbol definitions

- **Pin-definable Vs. Non Pin-definable**
  - Pin-definable
    - Used when each pin in a port can be individually programmed to act either as an input line or as an output line. The direction of each pin is defined in the port’s PxDDR
  - Not pin-definable
    - Used when ALL the bits in the port act as either input lines or as output lines

- **Latched Vs. Non-latched**
  - Latched
    - The value read by the CPU at the port reflects the state of the input pin at the moment it was latched (i.e., when the input strobe was asserted)
  - Non-latched
    - The value read by the CPU at the port reflects the state of the input pin at the moment it is read (i.e., instantaneous value)

- **Unidirectional Vs. Bidirectional**
  - Unidirectional (Modes 0 and 1)
    - The direction of data flow is determined by the PxDDR and can only be modified by reconfiguring this register
  - Bidirectional (Modes 2 and 3):
    - Data can flow in any direction and the contents of the PxDDR are ignored.

- **Primary Data Direction**
  - The direction of the data transfer that permits double-buffering

---

**LEGEND**

- **D0**
- **D7**
- **Inputs**
- **Outputs**
- **Pin-definable**
- **Input strobe**

- **Unidirectional**
- **Double-buffered**
- **Bidirectional**
- **Single-buffered**
- **Non-latched**

*NPd: This is the ONLY mode that is Not Pin-definable*
The PI/T’s modes of operation

**Mode 0**
- PGCR: 00
- **Unidirectional**
  - 8-bit mode
  - 16-bit mode

**Mode 1**
- PGCR: 01
- **Unidirectional**
  - 8-bit mode
  - 16-bit mode

**Mode 2**
- PGCR: 10
- **Bidirectional**
  - 8-bit mode

**Mode 3**
- PGCR: 11
- **Bidirectional**
  - 16-bit mode

**Submodes**
- **00** Pin-definable double-buffered input or single-buffered output
- **01** Pin-definable double-buffered output or non-latched input
- **1X** Pin-definable single-buffered output or non-latched input
- **X0** Pin-definable double-buffered input or single-buffered output
- **01** Pin-definable double-buffered output or non-latched input

Port A is pin-definable single-buffered output or non-latched input. Port B is double-buffered bidirectional I/O.

Ports A and B form a single bidirectional 16-bit port.
The PI/T’s modes of operation

<table>
<thead>
<tr>
<th>Mode 0 (1D/8bit)</th>
<th>Mode 1 (1D/16bit)</th>
<th>Mode 2 (2D/8bit)</th>
<th>Mode 3 (2D/16bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**LEGEND**
- **Unidirectional**
- **Double-buffered**
- **Bidirectional**
- **Single-buffered**
- **Non-latched**

*NPd*: This is the ONLY mode that is Not Pin-definable
Mode 0, sub-mode 00

- **Data flow**
  - Double-buffered input or
  - Single-buffered output

- **Applications**
  - Normally used to receive data from devices such as A/D converters

- **Handshaking**
  - Data is latched into the input register by the asserted edge of H1
  - H2 behaves according to its programming function defined below

- **Port B behaves identically (using H3 and H4)**

### Bit Configuration

<table>
<thead>
<tr>
<th>Bit</th>
<th>PACR7</th>
<th>PACR6</th>
<th>PACR5</th>
<th>PACR4</th>
<th>PACR3</th>
<th>PACR2</th>
<th>PACR1</th>
<th>PACR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-mode 00</td>
<td>H2 Control</td>
<td>H2 Int.</td>
<td>H1 Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PACR5</td>
<td>PACR4</td>
<td>PACR3</td>
<td>H2 Control</td>
<td>H2S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Input pin: Edge-sensitive input</td>
<td>Set on asserted edge</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Output pin: negated</td>
<td>Always clear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Output pin: asserted</td>
<td>Always clear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Output pin: interlocked input handshake</td>
<td>Always clear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Output pin: pulsed input handshake</td>
<td>Always clear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PACR2</th>
<th>H2 interrupt</th>
<th>PACR1</th>
<th>PACR0</th>
<th>H1 Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H2 interrupt disabled</td>
<td>X</td>
<td>X</td>
<td>H1 interrupt and DMA request disabled</td>
</tr>
<tr>
<td>1</td>
<td>H2 interrupt enabled</td>
<td>X</td>
<td>X</td>
<td>H1 interrupt and DMA request enabled</td>
</tr>
</tbody>
</table>

X | X | H1S status set anytime data is available in the double-buffered input path
Mode 0, submode 00: example

- Configure the PI/T in mode 0, sub-mode 00 where
  - Port A
    - Bits 7 and 6 of port A are used for output
    - Bits 5 to 0 of port A are used for input
  - Pulsed handshake is used in the primary direction
  - Handshake signals of Port A are active-high
  - Interrupts are disabled

```
PIT EQU $FE8001 ;PI/T base address on the SBC68K
PGCR EQU $00 ;offset of PGCR
PACR EQU $0C ;offset of PACG
PADDR EQU $04 ;offset of PADDR
PADR EQU $10 ;offset of PADR

setup LEA PIT,A0 ;A0 points to the PI/T’s base address
  MOVE.B #00000000, PGCR(A0) ;setup PGCR for mode 0 operation
  * MOVE.B #00111000, PACG(A0) ;setup port A for submode 00 operation
  * move.b #11000000, PADDR(A0) ;with pulsed H2 output handshake
  * move.b #00010011, PGCR(A0) ;enable H12 and make H1,H2 active-high
  * note that H12 is set to enable H1 and H2
RTS

send MOV.B D0,PADR(A0) ;write to port A to output bits 7 and 6
RTS

receive MOV.B PADR(A0),D0 ;read from port A to input bits 5 to 0
RTS
```
Mode 0, sub-mode 01

- **Data flow**
  - Double-buffered output or
  - Non-latched input

- **Applications**
  - Normally used to send data to devices such as D/A converters or printers

- **Tables are almost identical to 0/00 except for PACR0**
  - If PACR0=0, H1S is set when port A is half-empty
  - If PACR0=1, H1S is set when port A is full-empty

- **Port B control is identical (using H3 and H4, of course)**
Mode 0, sub-mode 1X

- **Data flow**
  - Single-buffered output or
  - Non-latched input

- **Applications**
  - A simple general-purpose bit I/O facility in which the various bits may be used individually to perform input or output as required

- **H1 control**
  - H1 is edge-sensitive and plays NO ROLE in any handshaking
  - H2 may be programmed as an edge-sensitive input that sets bit H2S when asserted

- **Port B behaves identically (using H3 and H4)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>PACR7</th>
<th>PACR6</th>
<th>PACR5</th>
<th>PACR4</th>
<th>PACR3</th>
<th>PACR2</th>
<th>PACR1</th>
<th>PACR0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>H2 Control</td>
<td></td>
<td></td>
<td>H2 Int.</td>
<td></td>
<td>H1 Control</td>
</tr>
</tbody>
</table>

**Sub-mode 1X**

<table>
<thead>
<tr>
<th>PACR5</th>
<th>PACR4</th>
<th>PACR3</th>
<th>H2 Control</th>
<th>H2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Input pin: Edge-sensitive input</td>
<td>Set on asserted edge</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>Output pin: negated</td>
<td>Always clear</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>1</td>
<td>Output pin: asserted</td>
<td>Always clear</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PACR2</th>
<th>H2 interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H2 interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>H2 interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PACR1</th>
<th>PACR0</th>
<th>H1 Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>H1 interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>H1 interrupt enabled</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>H1 is edge-sensitive input: H1S status set by the asserted edge of H1</td>
</tr>
</tbody>
</table>
Mode 1, sub-mode X0

- **Data flow**
  - Double-buffered input or
  - Single-buffered output

- **Port A should contain the Most Significant Byte in the word**
  - The MOVEP.W PADR(A0),DO instruction will behave as follows:
    - \([DO(8:15)] \leftarrow [PADR(A0)]\)
    - \([DO(0:7)] \leftarrow [PBDR(A0)]\)

- **The operation of the port is controlled by PBCR and (H3,H4)**
  - Port A Control Register used to provide additional facilities with signals H1 and H2

**Bit**

<table>
<thead>
<tr>
<th>PACR7</th>
<th>PACR6</th>
<th>PACR5</th>
<th>PACR4</th>
<th>PACR3</th>
<th>PACR2</th>
<th>PACR1</th>
<th>PACR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>H2 Control</td>
<td></td>
<td>H2 Int.</td>
<td></td>
<td>H1 Control</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>← Sub-mode XX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **PACR0-PACR5 behave exactly as in Mode 0, submode 1X**

**Bit**

<table>
<thead>
<tr>
<th>PBCR7</th>
<th>PBCR6</th>
<th>PBCR5</th>
<th>PBCR4</th>
<th>PBCR3</th>
<th>PBCR2</th>
<th>PBCR1</th>
<th>PBCR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>H4 Control</td>
<td></td>
<td>H4 Int.</td>
<td></td>
<td>H3 Control</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>← Sub-mode X0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**H4 Control**

<table>
<thead>
<tr>
<th>PBCR5</th>
<th>PBCR4</th>
<th>PBCR3</th>
<th>H4 Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Input pin: Edge-sensitive input  Set on asserted edge</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Output pin: negated  Always clear</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Output pin: asserted  Always clear</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Output pin: interlocked input handshake  Always clear</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Output pin: pulsed input handshake  Always clear</td>
</tr>
</tbody>
</table>

**H4 interrupt**

<table>
<thead>
<tr>
<th>PBCR2</th>
<th>H4 interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H4 interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>H4 interrupt enabled</td>
</tr>
</tbody>
</table>

**H3 Control**

<table>
<thead>
<tr>
<th>PBCR1</th>
<th>PBCR0</th>
<th>H3 Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>H3 interrupt and DMA request disabled</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>H3 interrupt and DMA request enabled</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>H3S status set if data present in double-buffered input path</td>
</tr>
</tbody>
</table>
Mode 1, sub-mode X1

- **Data flow**
  - Double-buffered output or
  - Non-latched input

- **Writing to the PI/T**
  - the MSB should be written to Port A and the LSB to port B IN THIS ORDER

- **The operation of the port is similar to Mode 1/X0 except for PBCR0**

---

<table>
<thead>
<tr>
<th>Bit</th>
<th>PACR7</th>
<th>PACR6</th>
<th>PACR5</th>
<th>PACR4</th>
<th>PACR3</th>
<th>PACR2</th>
<th>PACR1</th>
<th>PACR0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>H2 Control</td>
<td>H2 Int.</td>
<td>H1 Control</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sub-mode XX

PACR0-PACR5 behave exactly as in Mode 0, submode 1X

<table>
<thead>
<tr>
<th>Bit</th>
<th>PBCR7</th>
<th>PBCR6</th>
<th>PBCR5</th>
<th>PBCR4</th>
<th>PBCR3</th>
<th>PBCR2</th>
<th>PBCR1</th>
<th>PBCR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1</td>
<td></td>
<td>H4 Control</td>
<td>H4 Int.</td>
<td>H3 Control</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sub-mode X1

<table>
<thead>
<tr>
<th>PBCR5</th>
<th>PBCR4</th>
<th>PBCR3</th>
<th>H4 Control</th>
<th>H3S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Input pin: Edge-sensitive input</td>
<td>Set on asserted edge</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Output pin: negated</td>
<td>Always clear</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Output pin: asserted</td>
<td>Always clear</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Output pin: interlocked input handshake</td>
<td>Always clear</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Output pin: pulsed input handshake</td>
<td>Always clear</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PBCR2</th>
<th>H4 interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H4 interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>H4 interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PBCR1</th>
<th>PBCR0</th>
<th>H3 Control</th>
<th>H3S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>H3 interrupt and DMA request disabled</td>
<td>H3S status set whenever the initial or final output latches can accept new data and cleared if both latches are full</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>H3 interrupt and DMA request enabled</td>
<td>H3S status set when both initial or final output latch are empty and cleared if at least one latch is full</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>H3S status set whenever the initial or final output latches can accept new data and cleared if both latches are full</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>H3S status set when both initial or final output latch are empty and cleared if at least one latch is full</td>
<td></td>
</tr>
</tbody>
</table>
Mode 2

- **Data flow**
  - Port A is unidirectional (direction is determined by PADDR as usual)
    - non-latched input or
    - single-buffered output
  - Port B is bidirectional, double-buffered, non-pin addressable I/O (PBDDR is ignored)

- **All the handshake signals are associated with Port B**
  - H1 and H2 control output transfers
    - Data written by the CPU is passed to the peripheral when the latter asserts H1
  - H3 and H4 control input transfers
    - Data latched on the asserted edge of H3
Mode 3

- **Data flow**
  - Both ports act like a 16-bit bidirectional, double-buffered, non-pin addressable I/O port

- **Applications**
  - Relatively high speed data transfers
    - Keep in mind that the PI/T-CPU data bus is 8-bit wide so two R/W cycles are necessary

- **Handshake signals are similar to Mode 2**
  - H1 and H2 control output transfers
  - H3 and H4 control input transfers
An example in C language

- Write a C program to
  - Read data from Port A in a non-latched fashion
  - Write the data from Port A to Port B in a single-buffered fashion
  - This procedure should be performed periodically every 5 seconds
  - USE TIMER INTERRUPTS!!!
    - The vector number is 70 (decimal)
/* Timer Register Addresses */
#define tmr ((unsigned char*) 0xFE8021) /* Timer Base Address */
#define tcr  ((unsigned char*) tmr)     /* Timer Control Reg */
#define tivr ((unsigned char*) tmr+?)   /* Timer Interrupt Vector Reg */
#define cprh ((unsigned char*) tmr+?)   /* Preload Hi Reg */
#define cprm ((unsigned char*) tmr+?)   /* Preload Mid Reg */
#define cpfl ((unsigned char*) tmr+??)  /* Preload Lo Reg */
#define cnlh ((unsigned char*) tmr+??)  /* Counter Hi Reg */
#define cnlr ((unsigned char*) tmr+??)  /* Counter Lo Reg */
#define tshr ((unsigned char*) tmr+??)  /* Timer Status Reg */

/* Parallel I/O Register Addresses */
#define PGCR  (unsigned char*)0xFE80??   /* PI/T General Control Reg */
#define PSRR (unsigned char*)0xFE80??   /* PI/T Service Routine Reg */
#define PIVR (unsigned char*)0xFE80??   /* PI/T Interrupt Vector Reg */
#define PACR (unsigned char*)0xFE80??   /* PI/T Status Reg */
#define PADDR (unsigned char*)0xFE80??   /* Port A Data Direction Reg */
#define PADR (unsigned char*)0xFE80??   /* Port A Data Reg */
#define PBCHR (unsigned char*)0xFE80??   /* Port B Control Reg */
#define PBDDR (unsigned char*)0xFE80??   /* Port B Data Reg */
#define PBDR  (unsigned char*)0xFE80??   /* Port B Data Reg */

void isr() {
    printf("Five secs has passed\n");
    *pbdr = *padr ; /* This is really the main job of isr *
    It copies the content ports data register (our input port)
    and then places it to port B (our output port)*/
    *tsr = 0x01; /* reset the ZDS bit */
    asm(" rte");
}

main () {
    long *vtable;
    int count=7;
    asm(" move.w #$2400,SR");
    asm(" movea.l #$20000,SP");
    *PGCR = 0x??; /* disable Port A & B */
    *PADDR = 0x??; /* Set Port A as input */
    *PBCHR = 0x??; /* set Port B as output */
    *PSRR = 0x??; /* set PI/T for no interrupts */
    *PBCHR = 0x??; /* set Port B as output */
    *PACK = 0x??; /* set Port A as input */

    ****Prepare CPU for an interrupt processing****
    *tivr = ??;
    vtable = (long *) (??*??);
    *vtable = isr;

    ****Set up timer control register****
    *tcr = 0x??; /* Set Timer Mode */
    *cprh = (unsigned char) count;
    count = count >> 8; /* shift right 8 bits */
    *cprm = (unsigned char) count;
    count = count >> 8; /* shift right 8 bits */
    *cpfl = (unsigned char) count;
    *tsr = 0x??; /* Start timer */

    while (1) {
        /* Create an infinite loop which does nothing*/
    }
}
/* Timer Register Addresses */
#define tmr  ((unsigned char*) 0xFE8021) /* Timer Base Address */
#define tcr  (( unsigned char*) tmr)     /* Timer Control Reg */
#define tivr (( unsigned char*) tmr+2)   /* Timer Interrupt Vector Reg */
#define cprh (( unsigned char*) tmr+6)   /* Preload Hi Reg */
#define cprm (( unsigned char*) tmr+8)   /* Preload Mid Reg */
#define cplr (( unsigned char*) tmr+10) /* Preload Lo Reg */
#define cnhr (( unsigned char*) tmr+14) /* Counter Hi Reg */
#define cnrm (( unsigned char*) tmr+16) /* Counter Mid Reg */
#define cnrl (( unsigned char*) tmr+18) /* Counter Lo Reg */
#define tsr  (( unsigned char*) tmr+20)  /* Timer Status Reg */

/* Parallel I/O Register Addresses */
#define PGCR  ( unsigned char*)0xFE8001   /* PI/T General Control Reg */
#define PSRR  ( unsigned char*)0xFE8003   /* PI/T Service Routine Reg */
#define PIVR  ( unsigned char*)0xFE800B   /* PI/T ... PADR ( unsigned char*)0xFE8011   /* Port A Data Reg */
#define PBCR  ( unsigned char*)0xFE800F   /* Port B Control Reg */
#define PACR  ( unsigned char*)0xFE801B   /* PI/T Status Reg */
#define PADDR ( unsigned char*)0xFE8000    /* Port A Data Direction Reg */
#define PADR ( unsigned char*)0xFE8011    /* Port A Data Reg */
#define PBDR ( unsigned char*)0xFE8006    /* Port B Control Reg */
#define PBDDR ( unsigned char*)0xFE8007   /* Port B Data Direction Reg */
#define PBDR  ( unsigned char*)0xFE8013   /* Port B Data Reg */

void isr() {
    printf("Five secs has passed\n");
    *pbdr = *paddr ; /* This is really the main job of isr *
    It copies the content porta data register (our input port) and then places it to port B (our output port)*/
    *tsr = 0x01; /* reset the ZDS bit */
    asm(" rte");
}

main () {
    long *vtable;
    int count=1250000;
    asm(" move.w "$2400,SR");
    asm(" movea.l "$20000,SP);
    *PGCR = 0x0F; /* disable Port A & B */
    *PADDR = 0x00; /* Set Port A as input */
    *PBDR = 0xFF; /* set Port B as Output */
    *PSRR = 0x00; /* set PI/T for no Interrupts */
    *PBDR = 0x00; /*0r 0x80*/ /* Set Port B Control */
    *PACK = 0x40; /*0r 0x80*/ /* Set Port A Control */

    /*****Prepare CPU for an interrupt processing***/
    *tivr = 70;
    vtable = (long *) (70*4);
    *vtable = isr;

    /*****Set up timer control register*/
    *tcr = 0xA0; /* Set Timer Mode */
    *cplr = (unsigned char) count;
    count = count >> 8; /* shift right 8 bits */
    *cprm = (unsigned char) count;
    count = count >> 8; /* shift right 8 bits */
    *cnrm = (unsigned char) count;
    *tivr = 70; /* Start timer */

    while (1) {
        /* Create an infinite loop which does nothing*/
    }
}