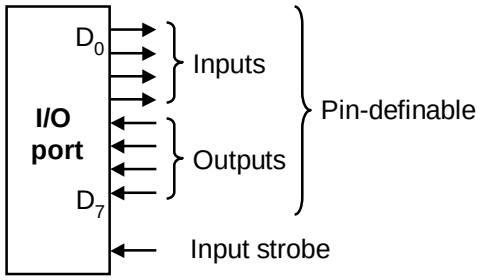
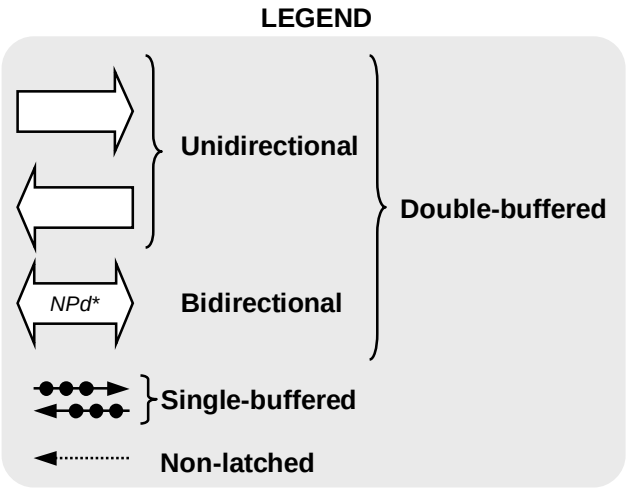
Lecture 12: PI/T parallel I/O, part II

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

Term and symbol definitions

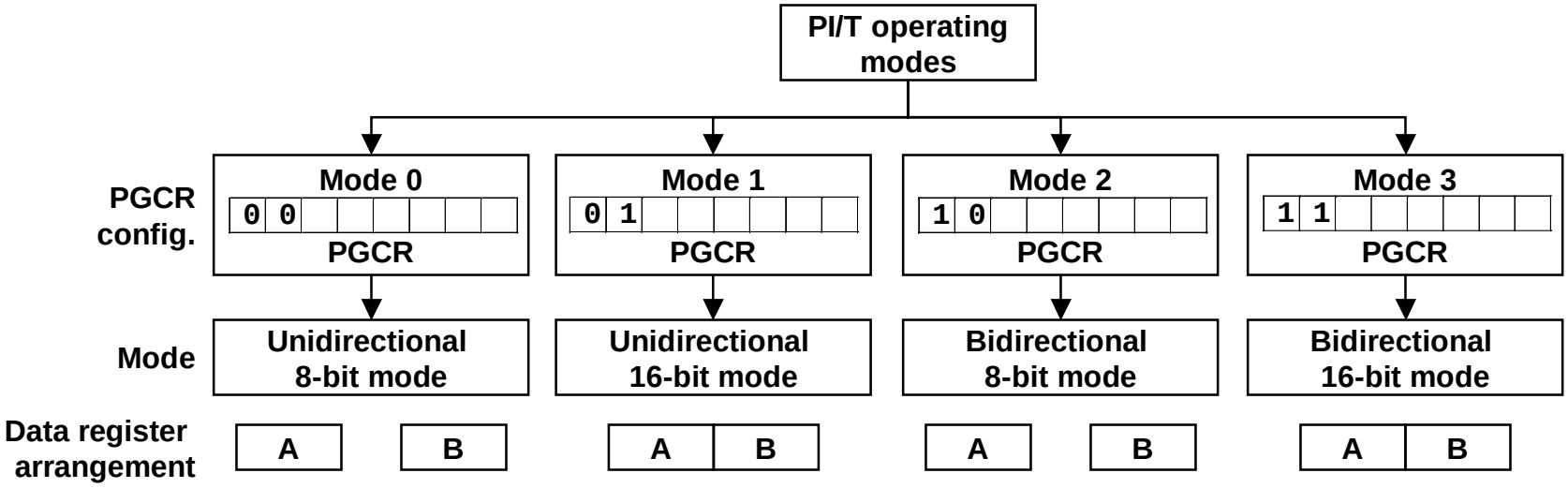
- **Pin-definable Vs. Non Pin-definable**
  - Pin-definable
    - Used when each pin in a port can be individually programmed to act either as an input line or as an output line. The direction of each pin is defined in the port's PxDDR
  - Not pin-definable
    - Used when ALL the bits in the port act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





**NPd: This is the ONLY mode that is Not Pin-definable*

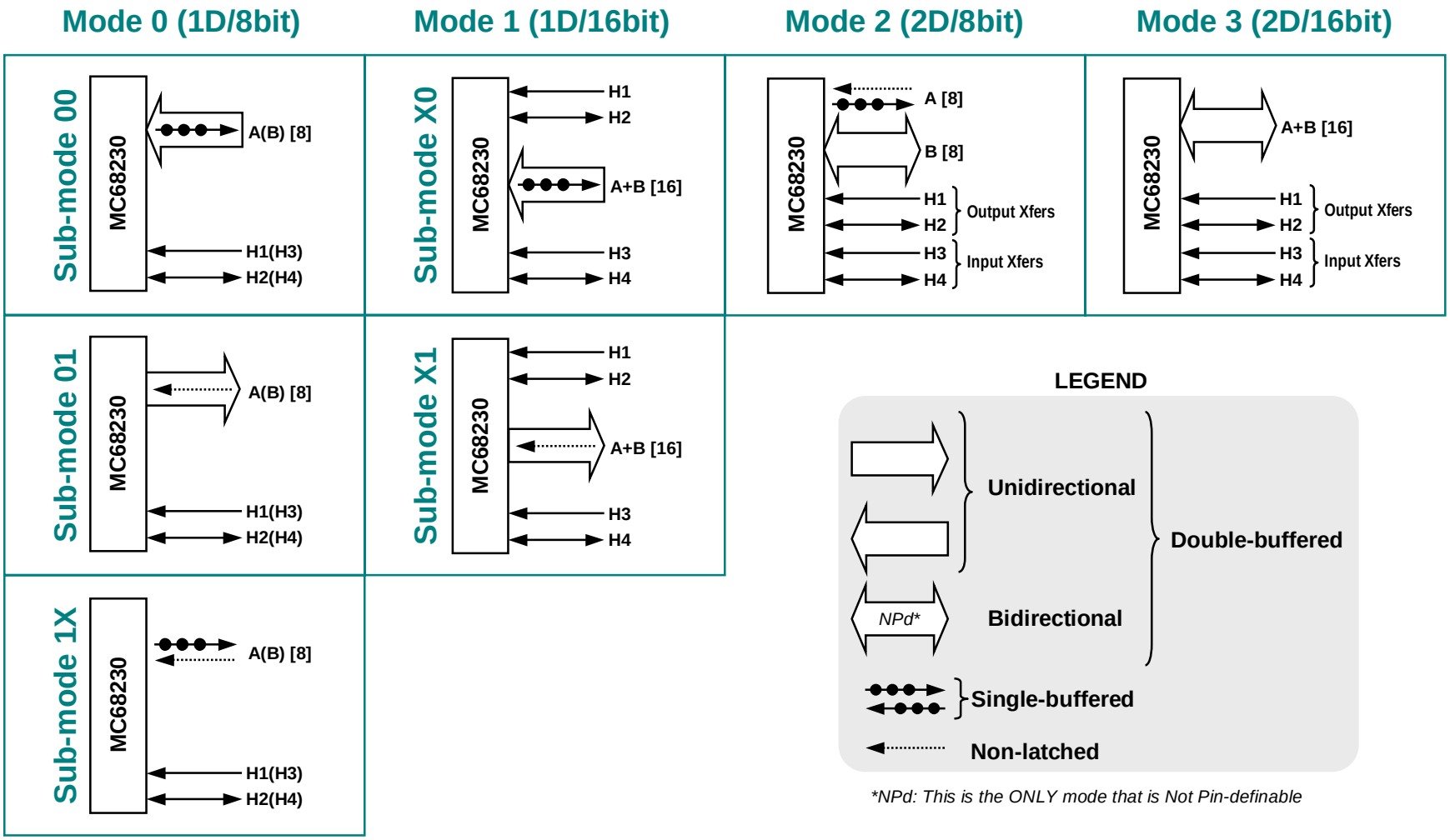
The PI/T's modes of operation



Submodes

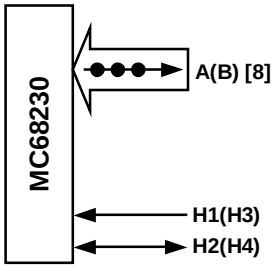
Mode 0	Mode 1	Mode 2	Mode 3
00 Pin-definable double-buffered input or single-buffered output	X0 Pin-definable double-buffered input or single-buffered output	Port A is pin-definable single-buffered output or non-latched input. Port B is double-buffered bidirectional I/O	Ports A and B form a single bidirectional 16-bit port
01 Pin-definable double-buffered output or non-latched input	01 Pin-definable double-buffered output or non-latched input		
1X Pin-definable single-buffered output or non-latched input			

The PI/T's modes of operation



*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	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	0	0	H2 Control			H2 Int.	H1 Control	

Sub-mode 00

PACR5	PACR4	PACR3	H2 Control	H2S
0	X	X	Input pin: Edge-sensitive input	Set on asserted edge
1	0	0	Output pin: negated	Always clear
1	0	1	Output pin: asserted	Always clear
1	1	0	Output pin: interlocked input handshake	Always clear
1	1	1	Output pin: pulsed input handshake	Always clear

PACR2	H2 interrupt
0	H2 interrupt disabled
1	H2 interrupt enabled

PACR1	PACR0	H1 Control
0	X	H1 interrupt and DMA request disabled
1	X	H1 interrupt and DMA request enabled
X	X	H1S status set anytime data is available in the double-buffered input path

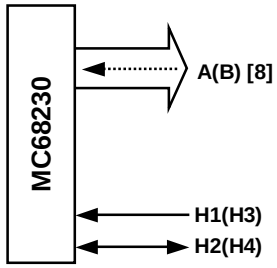
Mode 0, submode 00: example

- **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
*                                       ;with pulsed H2 output handshake
        MOVE.B  #%11000000, PADDR(A0)   ;setup bits 7,6 as outputs, 0 to 5 as inputs
        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    MOVE.B  D0,PADR(A0)             ;write to port A to output bits 7 and 6
        RTS
receive MOVE.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)**

Bit	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	0	1	H2 Control			H2 Int.	H1 Control	

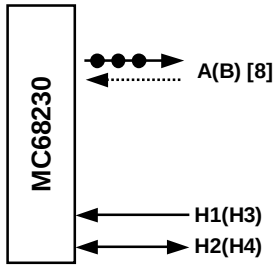
Sub-mode 01

PACR5	PACR4	PACR3	H2 Control	H2S
0	X	X	Input pin: Edge-sensitive input	Set on asserted edge
1	0	0	Output pin: negated	Always clear
1	0	1	Output pin: asserted	Always clear
1	1	0	Output pin: interlocked input handshake	Always clear
1	1	1	Output pin: pulsed input handshake	Always clear

PACR2	H2 interrupt
0	H2 interrupt disabled
1	H2 interrupt enabled

PACR1	PACR0	H1 Control
0	X	H1 interrupt and DMA request disabled
1	X	H1 interrupt and DMA request enabled
X	0	H1S status set if either initial or final output latches can accept data and cleared otherwise
X	1	H1S status set if both initial and final output latches are empty and cleared otherwise

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)**

Bit	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	1	X	H2 Control			H2 Int.	H1 Control	

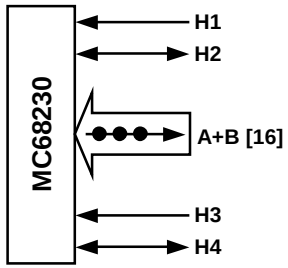
Sub-mode 1X

PACR5	PACR4	PACR3	H2 Control	H2S
0	X	X	Input pin: Edge-sensitive input	Set on asserted edge
1	X	0	Output pin: negated	Always clear
1	X	1	Output pin: asserted	Always clear

PACR2	H2 interrupt
0	H2 interrupt disabled
1	H2 interrupt enabled

PACR1	PACR0	H1 Control
0	X	H1 interrupt disabled
1	X	H1 interrupt enabled
X	X	H1 is edge-sensitive input: H1S status set by the asserted edge of H1

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)] ←[PADR(A0)]**
    - **[DO(0:7)] ←[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	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	0	0	H2 Control			H2 Int.	H1 Control	

Sub-mode XX — PACR0-PACR5 behave exactly as in Mode 0, submode 1X

Bit	PBCR7	PBCR6	PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0
	X	0	H4 Control			H4 Int.	H3 Control	

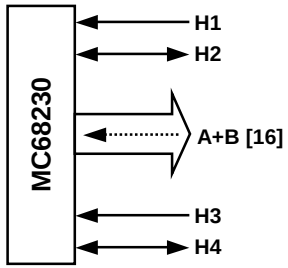
Sub-mode X0

PBCR5	PBCR4	PBCR3	H4 Control	H4S
0	X	X	Input pin: Edge-sensitive input	Set on asserted edge
1	0	0	Output pin: negated	Always clear
1	0	1	Output pin: asserted	Always clear
1	1	0	Output pin: interlocked input handshake	Always clear
1	1	1	Output pin: pulsed input handshake	Always clear

PBCR2	H4 interrupt
0	H4 interrupt disabled
1	H4 interrupt enabled

PBCR1	PBCR0	H3 Control
0	X	H3 interrupt and DMA request disabled
1	X	H3 interrupt and DMA request enabled
X	X	H3S status set if data present in double-buffered input path

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**

Bit	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	0	0	H2 Control			H2 Int.	H1 Control	
	Sub-mode XX		PACR0-PACR5 behave exactly as in Mode 0, submode 1X					

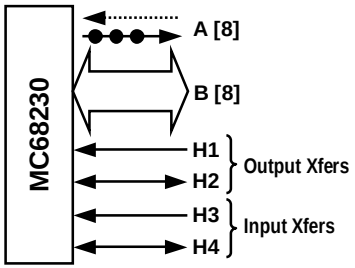
Bit	PBCR7	PBCR6	PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0
	X	1	H4 Control			H4 Int.	H3 Control	
	Sub-mode X1							

PBCR5	PBCR4	PBCR3	H4 Control	H4S
0	X	X	Input pin: Edge-sensitive input	Set on asserted edge
1	0	0	Output pin: negated	Always clear
1	0	1	Output pin: asserted	Always clear
1	1	0	Output pin: interlocked input handshake	Always clear
1	1	1	Output pin: pulsed input handshake	Always clear

PBCR2	H4 interrupt
0	H4 interrupt disabled
1	H4 interrupt enabled

PBCR1	PBCR0	H3 Control
0	X	H3 interrupt and DMA request disabled
1	X	H3 interrupt and DMA request enabled
X	0	H3S status set whenever the initial or final output latches can accept new data and cleared if both latches are full
X	1	H3S status set when BOTH the initial or final output latch are empty and cleared if at least one latch is full

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

Bit	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	X	X	H2 mode			H2 Interrupt	H1 Control	
	don't care							

PACR5	PACR4	PACR3	H2 interrupt	H2S
X	X	0	Output pin: Interlocked output handshake	Always cleared
X	X	1	Output pin: Pulsed output handshake	Always cleared

PACR2	H2 interrupt
0	H2 interrupt disabled
1	H2 interrupt enabled

PACR1	PACR0	H1 Control
0	X	H1 interrupt and DMA request disabled
1	X	H1 interrupt and DMA request enabled
X	0	H1S status set if initial or final output latches can accept data and cleared otherwise
X	1	H1S status set if both initial and final output latches are empty and cleared otherwise

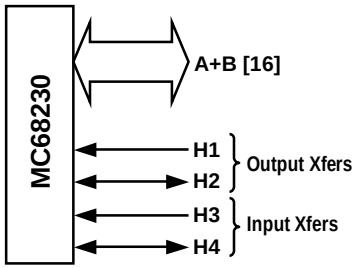
Bit	PBCR7	PBCR6	PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0
	X	X	H4 mode			H4 Interrupt	H3 Control	
	don't care							

PBCR5	PBCR4	PBCR3	H4 interrupt	H4S
X	X	0	Output pin: Interlocked output handshake	Always cleared
X	X	1	Output pin: Pulsed output handshake	Always cleared

PBCR2	H4 interrupt
0	H4 interrupt disabled
1	H4 interrupt enabled

PBCR1	PBCR0	H3 Control
0	X	H3 interrupt and DMA request disabled
1	X	H3 interrupt and DMA request enabled
X	X	H3S status set if data is available in the double-buffered input path

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

Bit	PACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	X	X	H2 mode			H2 Interrupt	H1 Control	
	don't care							

PACR5	PACR4	PACR3	H2 interrupt	H2S
X	X	0	Output pin: Interlocked output handshake	Always cleared
X	X	1	Output pin: Pulsed output handshake	Always cleared

PACR2	H2 interrupt
0	H2 interrupt disabled
1	H2 interrupt enabled

PACR1	PACR0	H1 Control
0	X	H1 interrupt and DMA request disabled
1	X	H1 interrupt and DMA request enabled
X	0	H1S status set if initial or final output latches can accept data and cleared otherwise
X	1	H1S status set if both initial and final output latches are empty and cleared otherwise

Bit	PBCR7	PBCR6	PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0
	X	X	H4 mode			H4 Interrupt	H3 Control	
	don't care							

PBCR5	PBCR4	PBCR3	H4 interrupt	H4S
X	X	0	Output pin: Interlocked output handshake	Always cleared
X	X	1	Output pin: Pulsed output handshake	Always cleared

PBCR2	H4 interrupt
0	H4 interrupt disabled
1	H4 interrupt enabled

PBCR1	PBCR0	H3 Control
0	X	H3 interrupt and DMA request disabled
1	X	H3 interrupt and DMA request enabled
X	X	H3S status set if data is available in the double-buffered input path

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)

Solution

```
/* 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 cprl (( unsigned char*) tmr+??)  /* Preload Lo Reg */
#define cnrh (( unsigned char*) tmr+??)  /* Counter Hi Reg */
#define cnrm (( unsigned char*) tmr+??)  /* Counter Mid Reg */
#define cnrl (( unsigned char*) tmr+??)  /* Counter Lo Reg */
#define tsr  (( 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 PSR   ( unsigned char*)0xFE80??   /* PI/T Status Reg */
#define PACR  ( unsigned char*)0xFE80??   /* PI/T Port A Control Reg */
#define PADDR ( unsigned char*)0xFE80??   /* Port A Data Direction Reg */
#define PADR  ( unsigned char*)0xFE80??   /* Port A Data Reg */
#define PBCR  ( unsigned char*)0xFE80??   /* Port B Control Reg */
#define PBDDR ( unsigned char*)0xFE80??   /* Port B Data Direction 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 porta data regsiter (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=?;

  asm("          move.w          #$2400,SR");
  asm("          movea.l         #$20000,SP);

  *PGCR =  0x??;                  /* disable Port A & B */
  *PADDR = 0x??;                  /* Set Port A as input */
  *PBDDR = 0x??;                  /* set Port B as Output  */
  *PSRR =  0x??;                  /* set PI/T for no Interrupts */
  *PBCR =  0x??; /*0r 0x??*/      /* Set Port B Control */
  *PACR =  0x??; /*0r 0x??*/      /* Set Port A Control */


  /****Prepare CPU for an interrupt processing**/

  *tivr   = ??;
  vtable  = (long *) (???*?);
  *vtable = isr;

  /****Set up timer control register*/

  *tcr = 0x??;                    /* Set Timer Mode */

  *cprl = (unsigned char) count;
  count = count >> 8;             /* shift right 8 bits */
  *cprm = (unsigned char) count;
  count = count >> 8;             /* shift right 8 bits */
  *cprh = (unsigned char) count;

  *tcr = 0x??;                    /* Start timer */

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

Solution

```
/* 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 cprl (( unsigned char*) tmr+10)  /* Preload Lo Reg */
#define cnrh (( 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 Interrupt Vector Reg */
#define PSR   ( unsigned char*)0xFE801B   /* PI/T Status Reg */
#define PACR  ( unsigned char*)0xFE800D   /* PI/T Port A Control Reg */
#define PADDR ( unsigned char*)0xFE8005   /* Port A Data Direction Reg */
#define PADR  ( unsigned char*)0xFE8011   /* Port A Data Reg */
#define PBCR  ( unsigned char*)0xFE800F   /* 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 = *padr ; /* This is really the main job of isr *
                   It copies the content porta data regsiter (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 */
  *PBDDR = 0xFF;                  /* set Port B as Output  */
  *PSRR =  0x00;                  /* set PI/T for no Interrupts */
  *PBCR =  0x00; /*0r 0x80*/      /* Set Port B Control */
  *PACR =  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 */

  *cprl = (unsigned char) count;
  count = count >> 8;             /* shift right 8 bits */
  *cprm = (unsigned char) count;
  count = count >> 8;             /* shift right 8 bits */
  *cprh = (unsigned char) count;

  *tcr = 0xA1;                    /* Start timer */


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