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 doublebuffering







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



The PI/T's modes of operation





The PI/T's modes of operation





Microprocessor-based System Design Ricardo Gutierrez-Osuna Wright State University

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 P	ACR7	PACR6	PAC	R5	PACR4	PACR3	PACR2	PACR1	PACR0					
	0	0			H2 Contro	bl	H1 Control							
		>												
Sub-mode 00														
PACR5	PA	CR4 P/	ACR3			H2 Contro		H2S						
0	-	X	Х	Inp	out pin: Edg	e-sensitive	Set on	Set on asserted edge						
1		0	0	Ou	tput pin: ne	egated	Alway	Always clear						
1		0	1	Ou	tput pin: as	serted	Alway	s clear						
1		1	0	Ou	tput pin: in	terlocked in	ke Alway	s clear						
1		1	1	Ou	tput pin: pu	ulsed input	Alway	s clear						
PACR2		12 interrup	t		PACR1	PACR0	H1 Cor	H1 Control						
0	H2 in	terrupt disa	abled	0 X H1 interrupt an					nd DMA request disabled					
1	H2 int	terrupt ena	bled		request enabled									

Х

Х

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 PGCR PACR PADDR PADR	EQU EQU EQU EQU EQU	\$FE8001 \$00 \$0C \$04 \$10	;PI/T b ;offset ;offset ;offset ;offset	ase address on the SBC68K of PGCR of PACG of PADDR of PADR
setup *	LEA MOVE.B MOVE.B MOVE.B MOVE.B	PIT,A0 #%00000000, #%00111000, #%11000000, #%00010011,	PGCR(A0) PACG(A0) PADDR(A0) PGCR(A0)	;A0 points to the PI/T's base address ;setup PGCR for mode 0 operation ;setup port A for submode 00 operation ;with pulsed H2 output handshake ;setup bits 7.6 as outputs, 0 to 5 as inputs ;enable H12 and make H1,H2 active-high ;note that H12 is set to enable H1 and H2
send	MOVE.B RTS	D0,PADR(A0)		;write to port A to output bits 7 and 6
receive	MOVE.B RTS	PADR(A0),D0		;read from port A to input bits 5 to 0



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 Contro	ĺ	H2 Int.	H1 C	ontrol			
		<i>~</i>	>									
		Sub-mo	ode 01									
	PAC	R5 PACE	R4 PACR3	3	H	2 Control			H2S			
	0	Х	X	Inpu	t pin: Edge-	sensitive i	input	Set on a	sserted edge			
	1	0	0	Outp	out pin: nega	Always	clear					
	1	0	1	Outp	out pin: asse	erted	Always	Always clear				
	1	1	0	Outp	out pin: inter	locked in	Always	clear				
	1	1	1	Outp	out pin: puls	ed input h	andshake	Always	clear			
PAC	20	LI2 in	torrupt		PACR1	PACRO		H1 Co	ontrol			
0	12	H2 interru	upt disabled		0	Х	H1 interrupt a	and DMA re	quest disabled			
1		H2 interr	upt enabled		1	Х	H1 interrupt a	and DMA re	quest enabled			
	I				v	0	H1S status se	et if either i	nitial or final ou			
					^	U	latches can accept data and cleared other					
					¥	1	H1S status se	et if both in	if both initial and final outpu			
					~	•	latches are e	mpty and c	leared otherwis			





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	Х		H2 Control		H2 Int.	t. H1 Control				
	Cub m										
	Sub-m	lode 1X									
	PACR5	PACR4	PACR3	H2	Control		S				
	0	X	Х	Input pin: Ed	lge-sensitiv	e input	input Set on asserted edge				
	1	х	0	Output pin: r	negated	•	Always clea	r			
	1	X	1	Output pin: a	asserted		Always clear				
ACR2		H2 interrup	t	PACR1	PACR0		H1 Co	ontrol			
0	H2 in	terrupt dis	abled	0	X	H1 inter	H1 interrupt disabled				
1	H2 interrupt enabled			1	Х	H1 interrupt enabled					
	_ _			X	х	H1 is ec	dge-sensitive	e input: H1S			
						000.090	110 00001100	eage et in			





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)] \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	ACR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0
	0	0		H2 Control		H2 Int.	H1 C	ontrol
~	Sub-mo	de XX	PACR	0-PACR5 be	ehave exac	tly as in Mo	de 0, submo	\rightarrow ode 1X
Bit P	BCR7	PBCR6	PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0
	Х	0		H4 Control		H4 Int.	H3 Co	ontrol
PBCR5	Sub-mo PBCR4	de X0 PBCR3			H4 Contro	1		
0	Х	Х	Input pin	: Edge-sens	sitive input			Set on asse
1	0	0	Output p	in: negated				Always clea
1	0	1	Output p	in: asserted	l			Always clea
	4	0	Output p	in: interlock	ed input h	andshake		Always clea
1		•						

H4 interrupt	PBCR1	PBCR0	H3 Control
H4 interrupt disabled	0	X	H3 interrupt and DMA request disabled
H4 interrupt enabled	1	Х	H3 interrupt and DMA request enabled
	Х	X	H3S status set if data present in double-buffered input path



0 1



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	PACR	PACR5	PACR4	PACR3	PACR2	PACR1	PACRO
	5.0	0	0		H2 Control	I / torto	H2 Int.	H1 C	ontrol
		~		→ ←			1		
		Sub-	mode XX	PACI	R0-PACR5 be	have exac	tly as in Mo	de 0, submo	ode 1X
	Bit	PBCR7	PBCR	B PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0
		Х	1		H4 Control		H4 Int.	H3 Co	ontrol
		←		>					
		Sub-	mode X1						
	PBCR5	PBCR4	PBCR3		H4 (Control			H4S
	0	Х	X	Input pin: E	dge-sensitive	e input		Set	on asserted ed
	1	0	0	Output pin:	negated	-		Alw	ays clear
	1	0	1	Output pin:	asserted			Alw	ays clear
	1	1	0	Output pin:	interlocked i	nput hand	shake	Alw	ays clear
	1	1	1	Output pin:	pulsed input	handshak	e	Alw	ays clear
-									
PBCR2		H4 interro	upt	PB	CR1 PBC	RO		H3 C	Control
0	H4 i	interrupt d	lisabled		0 X	H3 in	terrupt and	DMA reques	st disabled
1	H4	interrupt e	enabled		1 X	H3 in	terrupt and	DMA reque	st enabled
					X 0	H3S can a	status set w	henever the lata and cle	initial or final or final or final of a section of the section of
					V 4	H3S	status set w	hen BOTH t	he initial or fina

1

are empty and cleared if at least one latch is full

Х





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	Bit	PBCR7	Р	PBCR6	PBCR5	PBCR4	PBCR3	PBCR2 H4 Interrupt	PBCR1 H3 C	PBCR0
I	don'	t care	nz mode				don't care										
PAC X X	R5 PACR4 X X	PACR3 F 0 Output pin: Inter 1 Output pin: Puls	H2 interrupt locked output ha ed output hands	andshake hake	H2S Always cleared Always cleared			PBCF X X	R5 PBCI X X	R4 PB	3CR3 0 C 1 C	Dutput pin: Inte Dutput pin: Puls	H4 interrupt rlocked output l sed output hand	handshake Ishake	H4S Always cleared Always cleared		
PAC 0 1	PACR2 H2 interrupt 0 H2 interrupt disabled 1 H2 interrupt enabled														•		
PAC 0	R1 PACR0 X	H1 Contro H1 interrupt and DMA reque	ol est disabled					PBCF 0	1 PBCI X	10 H3	interrupt	H3 Control t and DMA requ	est disabled				
1 X	<u>x</u> 0	H1 interrupt and DMA reque H1S status set if initial or fi can accept data and cleared	<u>est enabled</u> nal output latche d otherwise	s				1 X	X	H3 H3S the	interrupt S status s double-l	<u>t and DMA requ</u> set if data is av buffered input i	est enabled ailable in path				
x	1	H1S status set if both initia latches are empty and clear	l and final outpur red otherwise	t					с.								



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	PA	CR7	PACR6	PACR5	PACR4	PACR3	PACR2	PACR1	PACR0	Bit	PBCR7	PB	CR6	PBCR5	PBCR4	PBCR3	PBCR2	PBCR1	PBCR0			
		Х	х		H2 mode		H2 Interrupt	H1 C	Control] [Х		Х		H4 mode		H4 Interrupt	H3 Co	ontrol			
	<i>~</i>		;	•							←		\longrightarrow									
		don't	care							don't care												
PAC	CR5	PACR4	PACR3		H2 interrupt		H2S			PBCF	5 PBC	R4 PBC	R3		H4 interrupt		H4S					
	<	Х	0	Output pin: Inte	rlocked output	handshake	Always cleared			X	×	0		Output pin: Inte	erlocked output	handshake	Always cleared					
)	(Х	1	Output pin: Puls	sed output hand	Ishake	Always cleared			X)	1		Output pin: Pul	sed output han	dshake	Always cleared		l			
PA0 (1 PA0	0 	H2 in H2 interru H2 interru PACR0	iterrupt pt disabled pt enabled	H1 Contro	ol					PBCF 0 1 PBCF	2 1 PBC	H4 Interrup H4 interrup H4 interrup	ot disal	bled bled H3 Control								
0)	Х	H1 interru	ot and DMA requ	est disabled					0	×	H3 in	nterrup	ot and DMA requ	lest disabled							
1		X	H1 interru	ot and DMA requ	est enabled					1	>	H3 in	nterrup	ot and DMA requ	lest enabled							
)	¢	0	H1S status can accept	s set if initial or f t data and cleare	inal output latcl d otherwise	nes				х	X	H3S s the d	status Iouble	set if data is av -buffered input	/ailable in path							
)	(1	H1S status latches are	s set if both initia empty and clea	al and final outp red otherwise	ut																



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 */
                                                                                   main () {
                                                                                     long *vtable;
#define tmr ((unsigned char*) 0xFE8021) /* Timer Base Address */
                                                                                     int count=?;
#define tcr (( unsigned char*) tmr)
                                        /* Timer Control Reg */
#define tivr (( unsigned char*) tmr+?) /* Timer Interrupt Vector Reg */
                                                                                     asm("
                                                                                                    move.w
                                                                                                                     #$2400,SR");
#define cprh (( unsigned char*) tmr+?) /* Preload Hi Reg */
                                                                                     asm("
                                                                                                    movea.l
                                                                                                                     #$20000,SP);
#define cprm (( unsigned char*) tmr+?) /* Preload Mid Reg */
#define cprl (( unsigned char*) tmr+??) /* Preload Lo Reg */
                                                                                     *PGCR = 0x??:
                                                                                                                     /* disable Port A & B */
                                                                                     *PADDR = 0x??;
                                                                                                                     /* Set Port A as input */
#define cnrh (( unsigned char*) tmr+??) /* Counter Hi Reg */
#define cnrm (( unsigned char*) tmr+??) /* Counter Mid Reg */
                                                                                     *PBDDR = 0x??;
                                                                                                                     /* set Port B as Output */
#define cnrl (( unsigned char*) tmr+??) /* Counter Lo Reg */
                                                                                     *PSRR = 0x??;
                                                                                                                     /* set PI/T for no Interrupts */
#define tsr (( unsigned char*) tmr+??) /* Timer Status Reg */
                                                                                     *PBCR = 0x??; /*0r 0x??*/
                                                                                                                     /* Set Port B Control */
                                                                                     *PACR = 0x??; /*0r 0x??*/
                                                                                                                     /* Set Port A Control */
/* Parallel I/O Register Addresses */
#define PGCR ( unsigned char*)0xFE80??
                                         /* PI/T General Control Reg */
                                                                                     /****Prepare CPU for an interrupt processing**/
#define PSRR ( unsigned char*)0xFE80??
                                         /* PI/T Service Routine Reg */
                                                                                     *tivr = ??;
#define PIVR ( unsigned char*)0xFE80??
                                         /* PI/T Interrupt Vector Reg */
                                                                                     vtable = (long *) (??*?);
#define PSR ( unsigned char*)0xFE80??
                                         /* PI/T Status Reg */
#define PACR ( unsigned char*)0xFE80??
                                         /* PI/T Port A Control Reg */
                                                                                     *vtable = isr;
#define PADDR ( unsigned char*)0xFE80??
                                         /* Port A Data Direction Reg */
#define PADR ( unsigned char*)0xFE80??
                                         /* Port A Data Reg */
                                                                                     /****Set up timer control register*/
#define PBCR ( unsigned char*)0xFE80??
                                         /* Port B Control Reg */
#define PBDDR ( unsigned char*)0xFE80??
                                         /* Port B Data Direction Reg */
                                                                                     *tcr = 0x??;
                                                                                                                     /* Set Timer Mode */
#define PBDR ( unsigned char*)0xFE80??
                                         /* Port B Data Reg */
                                                                                     *cprl = (unsigned char) count;
void isr() {
                                                                                     count = count >> 8;
                                                                                                                     /* shift right 8 bits */
 printf("Five secs has passed\n");
                                                                                     *cprm = (unsigned char) count;
                                                                                     count = count >> 8;
                                                                                                                     /* shift right 8 bits */
 *pbdr = *padr ; /* This is really the main job of isr *
                                                                                     *cprh = (unsigned char) count;
                 It copies the content porta data regsiter (our input port)
                 and then places it to port B (our output port)*/
                                                                                     *tcr = 0x??;
                                                                                                                     /* Start timer */
 *tsr = 0x01; /* reset the ZDS bit */
                                                                                     while (1) {
                                                                                       /* Create an infinite loop which does nothing*/
 asm(" rte");
}
                                                                                   }
```



Solution

```
/* Timer Register Addresses */
                                                                                   main () {
                                                                                     long *vtable;
#define tmr ((unsigned char*) 0xFE8021) /* Timer Base Address */
                                                                                     int count=1250000;
#define tcr (( unsigned char*) tmr)
                                        /* Timer Control Reg */
#define tivr (( unsigned char*) tmr+2)
                                       /* Timer Interrupt Vector Reg */
                                                                                     asm("
                                                                                                    move.w
                                                                                                                     #$2400,SR");
#define cprh (( unsigned char*) tmr+6) /* Preload Hi Reg */
                                                                                     asm("
                                                                                                    movea.l
                                                                                                                     #$20000,SP);
#define cprm (( unsigned char*) tmr+8) /* Preload Mid Reg */
#define cprl (( unsigned char*) tmr+10) /* Preload Lo Reg */
                                                                                     *PGCR = 0x0F:
                                                                                                                     /* disable Port A & B */
                                                                                     *PADDR = 0x00;
                                                                                                                     /* Set Port A as input */
#define cnrh (( unsigned char*) tmr+14) /* Counter Hi Reg */
#define cnrm (( unsigned char*) tmr+16) /* Counter Mid Reg */
                                                                                     *PBDDR = 0xFF;
                                                                                                                     /* set Port B as Output */
#define cnrl (( unsigned char*) tmr+18) /* Counter Lo Reg */
                                                                                     *PSRR = 0x00;
                                                                                                                     /* set PI/T for no Interrupts */
#define tsr (( unsigned char*) tmr+20) /* Timer Status Reg */
                                                                                     *PBCR = 0x00; /*0r 0x80*/
                                                                                                                     /* Set Port B Control */
                                                                                     *PACR = 0x40; /*0r 0x80*/
                                                                                                                     /* Set Port A Control */
/* Parallel I/O Register Addresses */
#define PGCR ( unsigned char*)0xFE8001
                                         /* PI/T General Control Reg */
                                                                                     /****Prepare CPU for an interrupt processing**/
#define PSRR ( unsigned char*)0xFE8003
                                         /* PI/T Service Routine Reg */
                                                                                     *tivr = 70;
#define PIVR ( unsigned char*)0xFE800B
                                         /* PI/T Interrupt Vector Reg */
                                                                                     vtable = (long *) (70*4);
#define PSR ( unsigned char*)0xFE801B
                                         /* PI/T Status Reg */
#define PACR ( unsigned char*)0xFE800D
                                         /* PI/T Port A Control Reg */
                                                                                     *vtable = isr;
#define PADDR ( unsigned char*)0xFE8005
                                         /* Port A Data Direction Reg */
#define PADR ( unsigned char*)0xFE8011
                                         /* Port A Data Reg */
                                                                                     /****Set up timer control register*/
#define PBCR ( unsigned char*)0xFE800F
                                         /* Port B Control Reg */
#define PBDDR ( unsigned char*)0xFE8007
                                         /* Port B Data Direction Reg */
                                                                                     *tcr = 0xA0;
                                                                                                                     /* Set Timer Mode */
#define PBDR ( unsigned char*)0xFE8013
                                         /* Port B Data Reg */
                                                                                     *cprl = (unsigned char) count;
void isr() {
                                                                                     count = count >> 8;
                                                                                                                     /* shift right 8 bits */
 printf("Five secs has passed\n");
                                                                                     *cprm = (unsigned char) count;
                                                                                     count = count >> 8;
                                                                                                                     /* shift right 8 bits */
 *pbdr = *padr ; /* This is really the main job of isr *
                                                                                     *cprh = (unsigned char) count;
                 It copies the content porta data regsiter (our input port)
                 and then places it to port B (our output port)*/
                                                                                     *tcr = 0xA1;
                                                                                                                     /* Start timer */
 *tsr = 0x01; /* reset the ZDS bit */
                                                                                     while (1) {
 asm(" rte");
                                                                                       /* Create an infinite loop which does nothing*/
}
                                                                                   }
```

