Tables of Contents

1. Introduction

- The TMR-10 is a general purpose counter/timer and digital I/O card
- PC AT compatible ISA bus
- On-board four 8254 chips
- 11 interrupt levels jumper selectable
- 4 different interrupt sources, 3 internal + 1 external, jumper selectable
- Flexible clock sources and gate control signals selectable
- 2 internal clock sources, 8M/1.6M, 0.8M/80K, jumper selectable
- 8 external clock sources (may be used as general purpose D/I, TTL level)
- 8 external gate control signals (may be used as general purpose D/I, TTL level)
- 8 bits general purpose D/O, TTL level
- 8 independent 16 bits timer/counter + 2 cascaded 32 bits timer/counter
- **All signals are TTL compatible**

1.1 Product Check List

In addition to this manual, the package includes the following items:

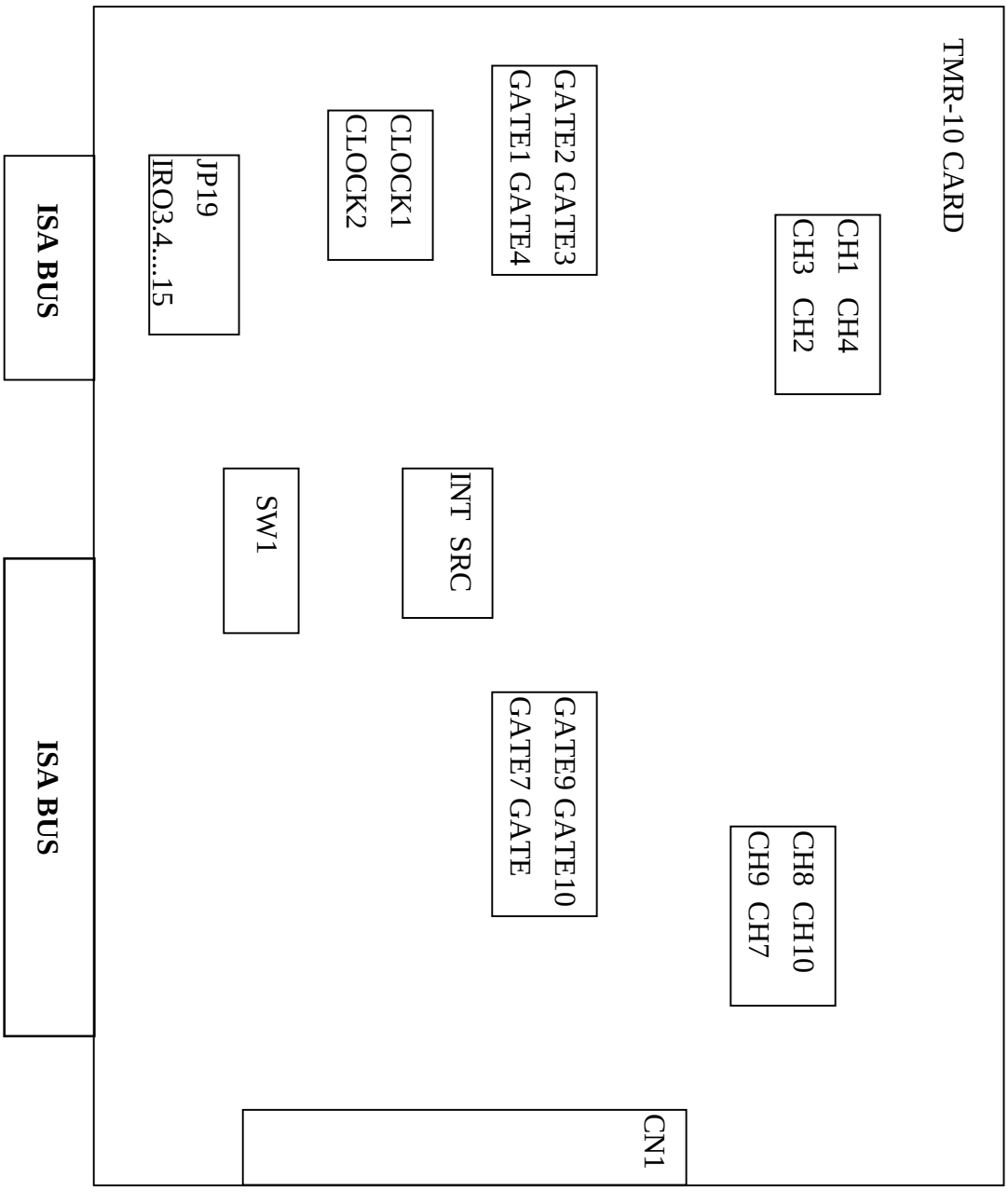
- TMR-10 card
- Demo program diskette

Attention !

If any of this items is missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton in case you want to ship or store the product in the future.

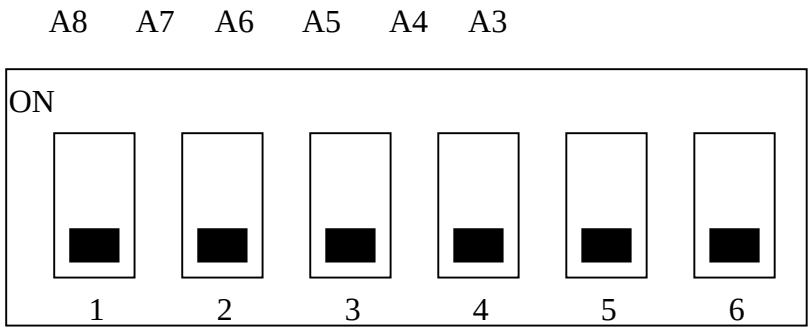
2. Hardware configuration

2.1 Board Layout



2.2 I/O Base Address Setting

The TMR-10 occupies 8 consecutive locations in I/O address space. The base address is set by DIP switch SW1. The default address is 0x200.



SW1 : BASE ADDRESS

BASE ADDR	A8	A7	A6	A5	A4	A3
✔ 200-207	OFF	OFF	OFF	OFF	OFF	OFF
208-20F	OFF	OFF	OFF	OFF	OFF	ON
210-217	OFF	OFF	OFF	OFF	ON	OFF
218-21F	OFF	OFF	OFF	OFF	ON	ON
220-227	OFF	OFF	OFF	ON	OFF	OFF
228-22F	OFF	OFF	OFF	ON	OFF	ON
230-237	OFF	OFF	OFF	ON	ON	OFF
238-23F	OFF	OFF	OFF	ON	ON	ON
:	:	:	:	:	:	:
:	:	:	:	:	:	:
3F0-3F7	ON	ON	ON	ON	ON	OFF
3F8-3FF	ON	ON	ON	ON	ON	ON

A9 is always equal to 1

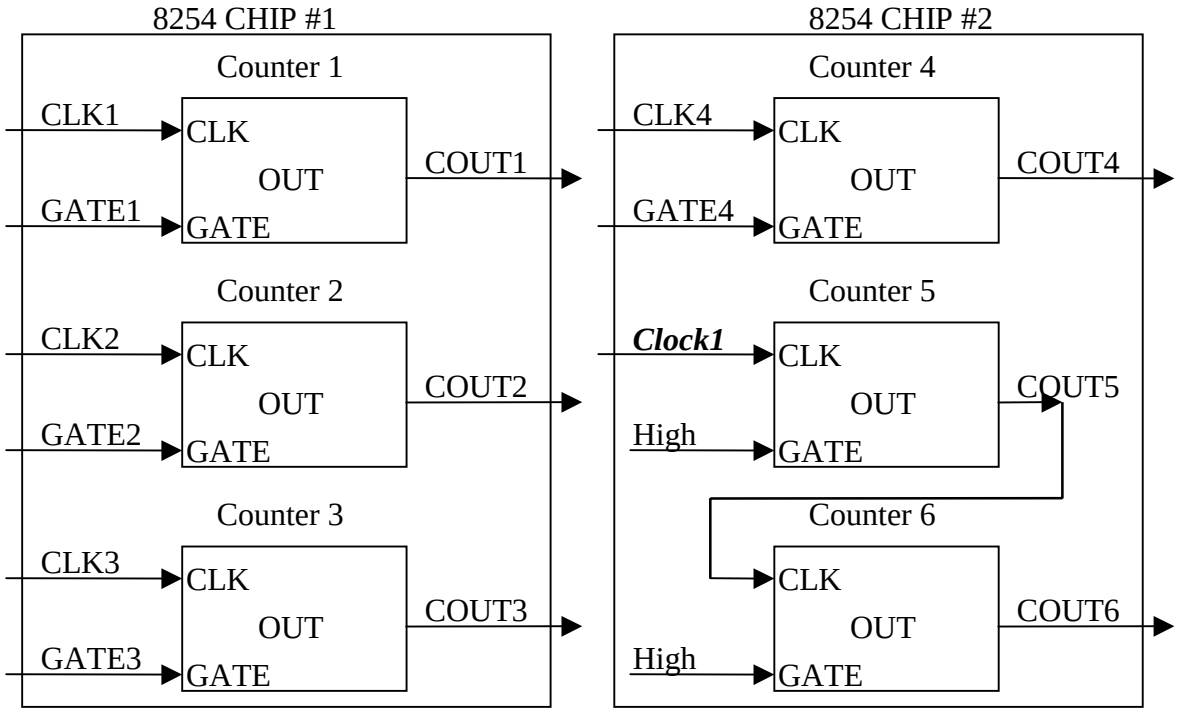
✔ Default Setting

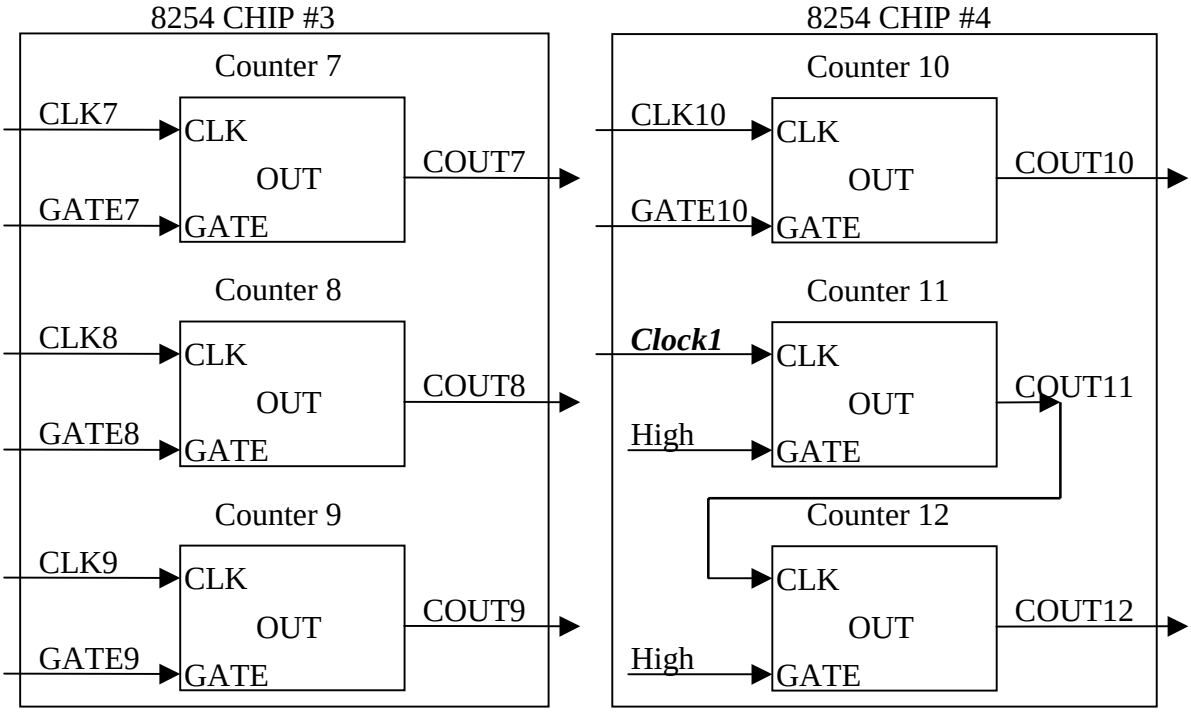
The PC I/O port mapping is given below.

I/O ADDRESS	Device
000-1FF	PC reserved
200-20F	Game controller
278-27F	LPT2
2F8-2FF	COM2
300-31F	Prototype card
320-32F	XT fixed disk
378-37F	LPT2
380-38F	SDLC
3A0-3AF	SDLE
3B0-3BF	Monochrome card
3C0-3CF	EGA card
3D0-3DF	CGA card
3F0-3FF	Diskette controller, COM1

2.3 Counter Architecture

There are four 8254 chips on the TMR-10 card. The block diagram is given below.



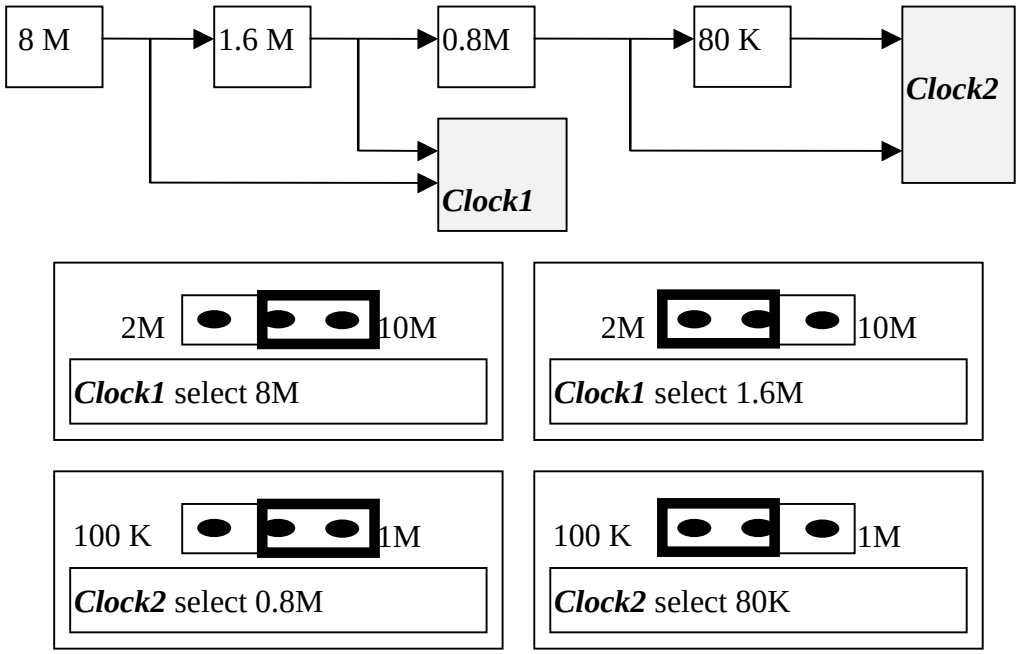


- 8 independent 16 bits timer/counter + 2 cascaded 32 bits timer/counter
- 8254 chip 1 --> counter 1 + counter 2 +counter 3
- 8254 chip 2 --> counter 4 + counter 5 +counter 6
- 8254 chip 3 --> counter 7 + counter 8 +counter 9
- 8254 chip 4 --> counter 10 + counter 11 +counter 12
- 8 independent 16 bits timer/counter --> counter 1,2,3,4,7,8,9,10
- 2 cascaded 32 bits timer/counter --> counter 5+6, counter 11+12
- CLK 1,2,3,4,7,8,9,10 refer to sec 2.4.3
- GATE 1,2,3,4,7,8,9,10 refer to sec 2.4.2
- ***Clock1*** refer to 2.4.1
- COUT 1,2,3,4,7,8,9,10 refer to sec 2.4.6
- GATE 5,6,11,12 always High
- CLK 5,11 always connect to ***Cock1***
- OUT5 connect to CLK6, OUT11 connect to CLK12

2.4 Jumper Setting

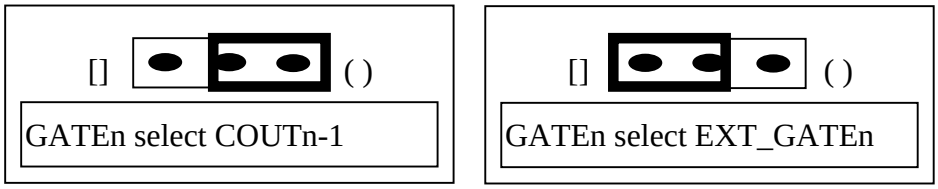
2.4.1 Clock1 & Clock2

There are two stable internal clock sources in TMR-10 card which named as ***clock1*** & ***clock2***. ***Clock1*** may be 8M/1.6M jumper selectable. ***Colck2*** may be 0.8M/80K jumper selectable. The block diagram of internal clock sources is given below:



2.4.2 Gate1,2,3,4,7,8,9,10

The gate control can be external signal or internal Cout comed from the last channel. For example, the GATE2 can be cascaded from Cout1, the GATE4 can be cascaded from Cout3.



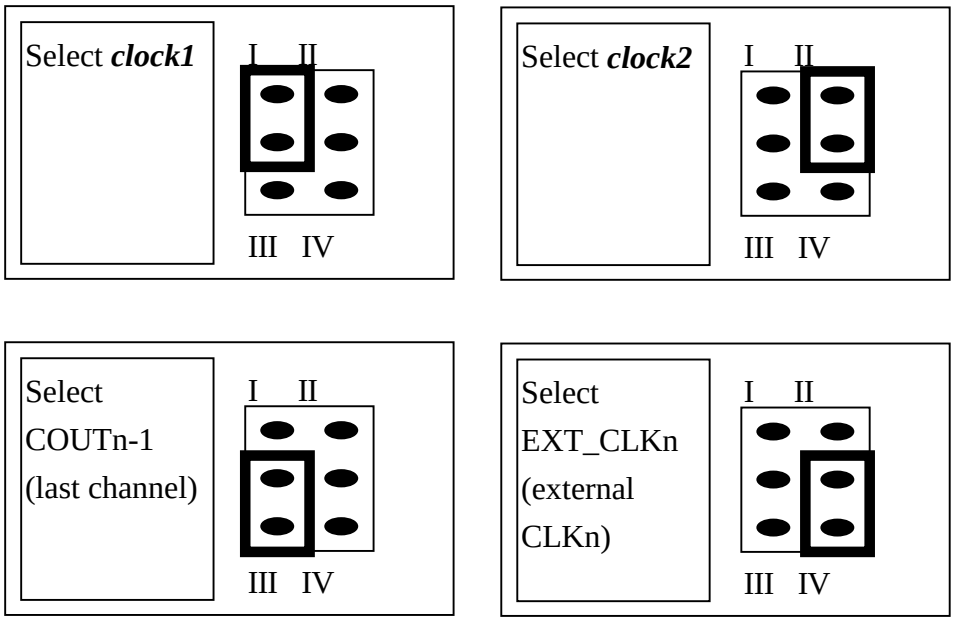
NOTE : GATE1 can select COUT4, GATE7 can select COUT10

2.4.3 CH 1,2,3,4,7,8,9,10

For counter 1,2,3,4,7,8,9,10, four signals can be selected as clock source. The counter 5+6 and the counter 11+12 use ***clock1*** as clock source.

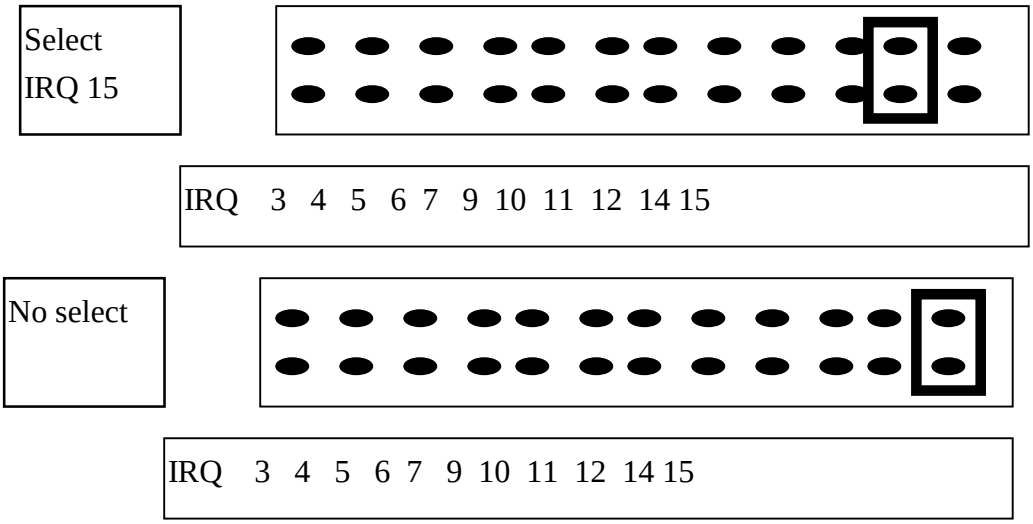
NOTE : CLK1 can select COUT4, CLK7 can selct COUT10

I : select clock1
II : select clock2
III : select COUTn-1, last channel COUT
IV : select ECLKn, external CLKn comed from CN1



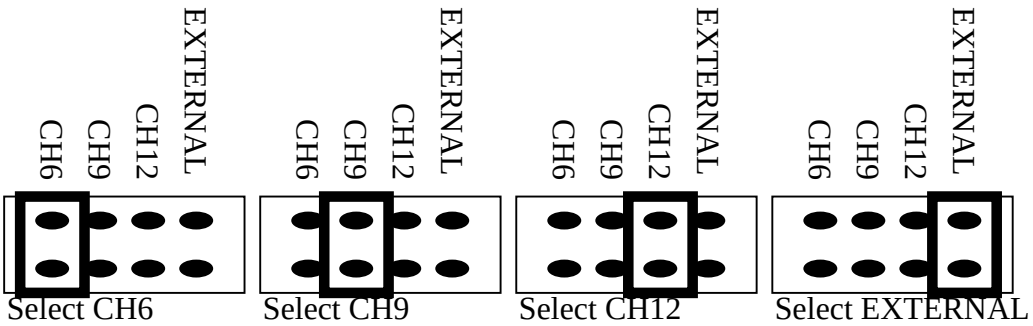
2.4.4 JP19, Interrupt Level Selection

The interrupt source comes from INT SRC jumper given in next section.



2.4.5 INT SRC, Interrupt Source Selection

There are four signals can be used as interrupt sources, CH6,CH9,CH12,EXTERNAL.
CH6 : comes from COUT6, output of cascaded counter 6
CH9 : comes from COUT9, output of independent counter 9
CH12 : comes from COUT12, output of cascaded counter12
EXTERNAL : comes from ECLK10, external CLK for counter 10, from CN1.



2.4.6 CN1 Pin Assignment

CN1 is a 37 pin D-type connector.

Pin Number	Description	Pin Number	Description
1	PCB‘s GND	20	PCB‘s GND
2	EXT_CLK1 / DI0	21	EXT_GATE1 / DI4
3	COUT1	22	EXT_GATE2 / DI5
4	EXT_CLK2 / DI1	23	COUT2
5	EXT_CLK3 / DI2	24	EXT_GATE3 / DI6
6	COUT3	25	EXT_GATE4 / DI7
7	EXT_CLK4 / DI3	26	COUT4
8	EXT_CLK7 / DI8	27	EXT_GATE7 / DI12
9	COUT7	28	EXT_GATE8 / DI13
10	EXT_CLK8 / DI9	29	COUT8
11	EXT_CLK9 / DI10	30	EXT_GATE9 / DI14
12	COUT9	31	EXT_GATE10 / DI15
13	EXT_CLK10 / DI11	32	COUT10
14	COUT6	33	COUT12
15	DO0	34	DO1
16	DO2	35	DO3
17	DO4	36	DO5
18	DO6	37	DO7
19	PCB‘s +5V output	XXXXXXX	This pin not available

EXT_CLKn : external clock source for counter n
EXT_GATEn : external gate control signal for counter n
COUTn : output of timer/counter n
DOn : digital output channel n
DIn : digital input channel n

All signals are TTL compatible.

2.5 I/O Register Address

The TMR-10 card occupies 8 consecutive PC I/O addresses. The following table lists the registers and their locations.

Address	Read	Write
Base+0	Active 8254 Counter 0	Active 8254 Counter 0
Base+1	Active 8254 Counter 1	Active 8254 Counter 1
Base+2	Active 8254 Counter 2	Active 8254 Counter 2
Base+3	Reserved	Active 8254 Counter Control
Base+4	Digital input channel 0-7 or External GATE 1-4, CLK 1-4	Select 8254 chip 1/2/3/4
Base+5	Digital input channel 8-15 or External GATE 7-10, CLK 7-10	Digital output channel 0-7

2.5.1 8254 Counter

There are four 8254 chips in TMR-10 card. Only one 8254 is active at a moment. Before using 8254, use BASE+4 to select the active 8254. The 8254 Programmable timer/counter has 4 registers from Base+0 through Base+3. For detailed programming information about 8254 , please refer to Intel‘s “Microsystem Components Handbook”.

Address	Read	Write
Base+0	Active 8254 Counter 0	Active 8254 Counter 0
Base+1	Active 8254 Counter 1	Active 8254 Counter 1
Base+2	Active 8254 Counter 2	Active 8254 Counter 2
Base+3	Reserved	Active 8254 Counter Control

2.5.2 Select 8254 chip 1/2/3/4

There are four 8254 chips in TMR-10 card. Only one 8254 is active at a moment. Before using 8254, use BASE+4 to select the active 8254.

(WRITE) Base+4 : select the active 8254 chip

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	X	X	D1	D0

D0=0, D1=0 : chip 1 8254 is active
D0=1, D1=0 : chip 2 8254 is active
D0=0, D1=1 : chip 3 8254 is active
D0=1, D1=1 : chip 4 8254 is active

2.5.3 Digital Input

(READ) Base+4 : read the external gate control signals and clock sources

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EXT_ GATE4	EXT_ GATE3	EXT_ GATE2	EXT_ GATE1	EXT_ CLK4	EXT_ CLK3	EXT_ CLK2	EXT_ CLK1

(READ) Base+4 : read the digital input channel 0 to 7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

(READ) Base+5: read the external gate control signals and clock sources

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EXT_ GATE10	EXT_ GATE9	EXT_ GATE8	EXT_ GATE7	EXT_ CLK10	EXT_ CLK9	EXT_ CLK8	EXT_ CLK7

(READ) Base+5: read the digital input channel 8to 15

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

The external gate control signals and clock sources can be read form BASE+4 and BASE+5. These pins can be used as digital input signals dedicatedly. Even if these pins are used, user also can read their signal level from BASE+4 & BASE+5.

2.5.4 Digital Output

(WRITE) Base+5 : set the digital output channel 0 to 7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

2.5.5 Factory Setting

- SW1 --> all in OFF position --> BASE address =0x200
- CH1,2,3,4,7,8,9,10 --> select ***Clock1***
- GATE 1,2,3,4,7,8,9,10 --> select [] --> external gate control
- ***Clock1*** --> select 8 M
- ***Clock2*** --> select 0.8 M
- INT SRC --> select CH 9
- JP19 --> select NO IRQ

3. Demo Programs

Software environment :

- Only three types of command, digital input, digital output, 8254 control command
- Multi-configurations
- Flexible clock sources and gate control signals
- To cascade multiple 8254 is available
- Suit for a variety of powerful timer/counter function

So we don‘t support any software drivers. We gives several demo program to show how to use the TMR-10 card. The demo program also show some application examples. These examples will help user to solve their real world problems without needing any software drivers.

The common features for these demo program are given below :

- Use Turbo C 2.XX compiler
- Source file : DEMO_??.C
- Project file : DEMO_??.PRJ
- Execution file : DEMO_??.exe

3.1 Use Digital Output

- Base address=0x200
- Initial send 0 to DO
- delay 0.3 second
- send 1 to DO
- delay 0.3 second
- send 2 to DO
- .
- .
- .
- delay 0.3 second
- send 0xff to DO
- delay 0.3 second
- send 0 to DO
- delay 0.3 second
- send 1 to DO
- .
- .
- .IO address of DO = BASE+5
- TTL level compatible
- ..

```
/*---------------------------------------------------------------------*/
/* filename=demo_01.c                                                  */
/*                                                                     */
/* test the output port                                                */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position     */
/*---------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>

#define  BASE 0x200          /* IO BASE address = 0x200 */

void main( void )
{
int DO;

DO=0;
for (;;)
      {
      outportb(BASE+5,DO);  /* send a value to output port */
      printf("\nDEMO_01.EXE --> DO=%d, press any key to stop",DO);
      delay(300);           /* delay 0.3 second */
      if (kbhit()!=0) {getch(); return;}
      DO=DO+1;              /* UP counter */
      DO=DO&0xff;           /* 8 bit counter */
      }
}


_
```

3.2 Use Digital Input

- IO address of DO = BASE+5
- IO address of DI = BASE+4 & BASE+5
- connect DO.0 to DI.0 & DI.8
- connect DO.1 to DI.1 & DI.9
- .
- .
- .
- connect DO.7 to DI.7 & DI..15

- send output value to DO
- read input value from DI
- DI0..7=DI8..15=DO0..7

```
/*---------------------------------------------------------------------*/
/* filename=demo_02.c                                                  */
/*                                                                     */
/* test the input  port                                                */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position     */
/* CN1.15 connect to CN1.2  & CN1.8   --> DO.0 TO DI1.0 & DI2.0        */
/* CN1.34 connect to CN1.4  & CN1.10  --> DO.1 TO DI1.1 & DI2.1        */
/* CN1.16 connect to CN1.5  & CN1.11  --> DO.2 TO DI1.2 & DI2.2        */
/* CN1.35 connect to CN1.7  & CN1.13  --> DO.3 TO DI1.3 & DI2.3        */
/* CN1.17 connect to CN1.21 & CN1.27  --> DO.4 TO DI1.4 & DI2.4        */
/* CN1.36 connect to CN1.22 & CN1.28  --> DO.5 TO DI1.5 & DI2.5        */
/* CN1.18 connect to CN1.24 & CN1.30  --> DO.6 TO DI1.6 & DI2.6        */
/* CN1.37 connect to CN1.25 & CN1.31  --> DO.7 TO DI1.7 & DI2.7        */
/*---------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#define  BASE 0x200           /* IO BASE address = 0x200 */

void main( void )
{
int DO,DI1,DI2;
DO=0;
for (;;)
      {
      outportb(BASE+5,DO);  /* send a value to output port */
      printf("\nDEMO_02.EXE --> DO=%2x,",DO);
      DI1=inportb(BASE+4);  /* digital input channel 0 to 7 */
      DI2=inportb(BASE+5);  /* digital input channel 8 to 15 */
      printf("DI1=%2x,DI2=%2x,press any key to stop",DI1,DI2);
      delay(300);           /* delay 0.3 second */
      if (kbhit()!=0) {getch(); return;}
      DO=DO+1;              /* UP counter */
      DO=DO&0xff;           /* 8 bit counter */
      }
}
```

3.3 Square Wave Generator

- counter 1 = single counter, located in 8254 chip 1
- counter 5+6 = cascaded counter, located in 8254 chip 2
- counter 7+8+9 = multi-counter, located in 8254 chip 3

- ***clock1*** select 8 M, ***clock2 s***elect 80K

- select 8254 chip 1
- program counter 1
- CH1 select ***clock2***=80K --> COUT1=80K/10=8K, square wave output

- select 8254 chip 2
- program counter 5 & counter 6
- CLK5=***clock1***=8M --> COUT5=8M/100=80K, square wave output
- CLK6=COUT5=80K --> COUT6=80K/80=1K, square wave output

- select 8254 chip 3
- program to counter 7 & counter 8 & counter 9
- CH7 select ***clock1***, CH8 select COUT7, CH9 select COUT8
- CLK7=***clock1***=8M --> COUT7=8M/100=80K, square wave output
- CLK8=COUT7=80K --> COUT8=80K/80=1K, square wave output
- CLK9=COUT8=1K --> COUT9=1K/100=10, square wave output

```
/*-------------------------------------------------------------------------------*/
/* filename=demo_03.c                                                            */
/* Square wave generator                                                         */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position               */
/* CLOCK1 select 8M,  CLOCK2 select 80K                                          */
/* CH1 select CLOCK2, COUT1=80K/10=8K --> single counter                         */
/* COUT5=8M/100=80K,COUT6=80K/80=1K --> cascaded counter                         */
/* CH7 select CLOCK1, CH8 select COUT7, CH9 select COUT8                         */
/* COUT7=8M/100=80K,COUT8=80K/80=1K,COUT9=1K/100=10 -->multi-counter
/*-------------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#define  BASE 0x200            /* IO BASE address = 0x200 */
void main(void)
{
/* ---------------- single counter ------------------------------------*/
outportb(BASE+4,0);     /* select 8254 chip_1 --> counter_1,2,3 */
/* CLK1=CLOCK2=80K --> COUT1=CLK1/10=8K */
outportb(BASE+3,0x36);       /* mode_3, counter_0 */
outportb(BASE,10);       /* counter_0 low byte first */
outportb(BASE,0);  /* counter_0 high byte */
/* ---------------- cascaded counter ----------------------------------*/
outportb(BASE+4,1);     /* select 8254 chip_2 --> counter_4,5,6 */
/* CLK5=CLOCK1=8M  --> COUT5=CLK5/100=80K */
outportb(BASE+3,0x76);       /* mode_3, counter_5 */
outportb(BASE+1,100); /* counter_5 low byte first */
outportb(BASE+1,0);     /* counter_5 high byte */
/* CLK6=COUT5=80K  --> COUT6=CLK6/80=1K */
outportb(BASE+3,0xb6);       /* mode_3, counter_6 */
outportb(BASE+2,80);   /* counter_6 low byte first */
outportb(BASE+2,0);     /* counter_6 high byte */

/* ---------------- multiple counter ----------------------------------*/
outportb(BASE+4,2);     /* select 8254 chip_3 --> counter_7,8,9 */
/* CLK7=CLOCK1=8M  --> COUT7=CLK7/100=80K */
outportb(BASE+3,0x36);       /* mode_3, counter_7 */
outportb(BASE,100);     /* counter_7 low byte first */
outportb(BASE,0);  /* counter_7 high byte */
/* CLK8=COUT7=80K --> COUT8=CLK8/80=1K */
outportb(BASE+3,0x76);       /* mode_3, counter_8 */
outportb(BASE+1,80);   /* counter_8 low byte first */
outportb(BASE+1,0);     /* counter_8 high byte */
/* CLK9=COUT8=1K --> COUT9=CLK9/100=10 */
outportb(BASE+3,0xb6);       /* mode_3, counter_9 */
outportb(BASE+2,100); /* counter_9 low byte first */
outportb(BASE+2,0);     /* counter_9 high byte */
}
_
```

3.4 Delay One Ms

- machine independent ***TIMER***

- ***clock1*** select 8M
- CH1 select ***clock1***=8M --> count 8000 = 1 ms
- counter 1 down count from 8000 --> 7999 --> 7998 ..... --> 1 --> 0 --> 0xffff --> 0xfffe .....

program detect counter1 = 0xff?? --> timer delay 1 ms

```
      /*---------------------------------------------------------------------*/
/* filename=demo_04.c                                                  */
/*                                                                     */
/* delay one ms                                                        */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position     */
/* CH1 select CLOCK1=8M --> count 8000 = 1 ms                          */
/*---------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#define  BASE 0x200            /* IO BASE address = 0x200 */

void main(void)
{
int i;
for (;;)
      {
      printf("\npress any key to stop\7");
      for (i=0; i<1000; i++) delay_one_ms(); /* delay 1000*1ms=1second */
      if (kbhit()!=0) {getch(); return;}
      }
}

/* ---------------------------------------------------------------------*/
delay_one_ms()
{
int high,low;
outportb(BASE+4,0);     /* select 8254 chip_1 --> counter_1,2,3 */
/* CLK1=CLOCK1=8M --> count 8000 = 1 ms */
outportb(BASE+3,0x30);       /* mode_0, counter_0 */
outportb(BASE,0x40);    /* counter_0 low byte first */
outportb(BASE,0x1F);    /* counter_0 high byte ,0x1F40=8000 */

for (;;)
      {
      outportb(BASE+3,0x00); /* latch counter_0 */
      low=inportb(BASE);
      high=inportb(BASE);
      if (high==0xff) return;
      }
}
```

3.5 16 Bits Event Counter

- Event down counter

- Initial load oxffff into counter
- The starting two event CLK will be used to initial 8254 chips
- **Event = 0xffff-current counter value+2**

```
/*-------------------------------------------------------------------------------*/
/* filename=demo_05.c                                                            */
/*                                                                               */
/* 16 bits event counter                                                         */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position               */
/* NOTE : (1) The 8254 need 2 CLKs to write initial counter values               */
/*      (2) The starting two event CLKs will be used to initialize 8254          */
/*      (3) Before initialization, the event counters holds their old        */
/*          value.                                                           */
/*-------------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>

#define  BASE 0x200            /* IO BASE address = 0x200 */

void main(void)
{
int i,high,low;
unsigned int event_counter;
outportb(BASE+4,0);     /* select 8254 chip_1 --> counter_1,2,3 */
/* CLK1=EXTERNAL CLOCK */
outportb(BASE+3,0x30);       /* mode_0, counter_0 */
outportb(BASE,0xFF);   /* counter_0 low byte first */
outportb(BASE,0xFF);   /* counter_0 high byte */
/* WARNING : Before the starting two event CLKS come, the event_counters
          hold their old value */
for (;;)
      {
      outportb(BASE+3,0x00);       /* latch counter_0 */
      low=inportb(BASE)&0xff;
      high=inportb(BASE)&0xff;
      event_counter=(0xFF-high)*256+(0xFF-low)+2;
                                   /* plus the starting two CLK */

      printf("\nevent_counter=%u --> high=%x,low=%x",
            event_counter,high&0xff,low&0xff);
      delay(500);
      if (kbhit()!=0) {getch(); return;}
      }
}
_
```

3.6 Software Event Counter

- Event down counter

- Initial load oxffff into counter
- The starting two event CLK will be used to initial 8254 chips
- counter down count from 0xffff --> 0xfffe -->..... --> 1 --> 0 --> 0xffff --> 0xfffe .....
-
-
- c65536 initial 0
- program detect counter = 0xff?? --> more 65536 event count --> c65536++
- **Event=65536*c65536+0xffff-current counter value+2**

```
/*-------------------------------------------------------------------------------*/
/* filename=demo_06.c                                                            */
/*                                                                               */
/* software event counter                                                        */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position               */
/* NOTE : (1) The 8254 need 2 CLKs to write initial counter values               */
/*      (2) The starting two event CLKs will be used to initialize 8254          */
/*      (3) Before initialization, the event counters holds their old        */
/*          value.                                                               */
/*-------------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#define  BASE 0x200            /* IO BASE address = 0x200 */
void main(void)
{
int i,high,low;
float event_counter,c65536;
char s0;
c65536=0.0; s0=0;
outportb(BASE+4,0);     /* select 8254 chip_1 --> counter_1,2,3 */
/* CLK1=EXTERNAL CLOCK */
outportb(BASE+3,0x30);       /* mode_0, counter_0 */
outportb(BASE,0xFF);   /* counter_0 low byte first */
outportb(BASE,0xFF);   /* counter_0 high byte */

/* WARNING : Before the starting two event CLKS come, the event_counters
           hold their old value */
for (;;)
      {
      outportb(BASE+3,0x00);       /* latch counter_0 */
      low=inportb(BASE)&0xff;
      high=inportb(BASE)&0xff;
      if (high==0) s0=1;
      if ((high==0xff) & (s0==1))
            {
            c65536+=1.0;
            s0=0;
            }
      event_counter=c65536*65536.0+(0xFF-high)*256.0+(0xFF-low)+2;
                                 /* plus the starting two CLK */
      printf("\nevent_counter=%f,high=%x,low=%x",
            event_counter,high&0xff,low&0xff);
      delay(500);
      if (kbhit()!=0) {getch(); return;}
      }
```

3.7 Watch Dog Timer

- ***Clock2*** select 80K
- IRQ level select IRQ5
- IRQ source select CH9=COUT9
- CH9 select ***clock2***=80K --> COUT9=80K/65535=1.2 Hz (about)
- Watchdog time interval=1.2Hz = 0.82 sec (about)

- The normal program will reflesh watchdog timer before watchdog timeout
- If program fail, the watchdog timer will interrupt CPU through IRQ5 after 0.82 sec

```
/*---------------------------------------------------------------------------*/
/* filename=demo_07.c                                                        */
/*                                                                           */
/* watch dog timer                                                           */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position           */
/* CH9 select CLOCK2=80K --> 80K/65535 = 1.2 Hz                              */
/*                  --> Watchdog time interval = 0.82 sec (about)            */
/* IRQ level  select IRQ5                                                    */
/* IRQ source select CH9=COUT9                                               */
/* COUT9 can trigger another hardware to perform hardware-protection         */
/*---------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

#define  BASE 0x200            /* IO BASE address = 0x200 */

static void   interrupt irq5_service();
int watchdog;

void main(void)
{
int i;

watchdog=0;
init_watchdog();

disable();
setvect (5+8,irq5_service);
outportb (0x21,inportb(0x21) & 0xdf);  /* 8259a irq5 enable */
enable();

for (;;)
      {
      reflesh_watchdog();       /* reflesh cycle must small than 0.82 sec */

      /* normal program run here */

      printf("\npress any key to simulate watchdog failure,watchdog=%d",watchdog);
      if (kbhit()!=0) {getch(); goto wait_watchdog;}
      }
wait_watchdog:
/* simulate program fail --> no reflesh_watchdog --> IRQ will interrupt CPU */
for(;;)
      {
      printf(" Wait");
      if (watchdog==1)
            {
```

```
            disable();
            outportb (0x21,inportb(0x21) | 0x20); /* 8259a irq5 disable */
            enable();
            printf("\nWatchdog failure");
            return;
            }
      if (kbhit()!=0) {getch(); return;}
      }
}

/* ---------------------------------------------------------------------*/
init_watchdog()
{
outportb(BASE+4,2);     /* select 8254 chip_3 --> counter_7,8,9 */
outportb(BASE+3,0xb0);       /* mode_0, counter_9 */
outportb(BASE+2,0xff); /* counter_9 low byte first */
outportb(BASE+2,0xff); /* counter_9 high byte ,0xffff=65535 */
}
/* ---------------------------------------------------------------------*/
reflesh_watchdog()
{
outportb(BASE+3,0xb0);       /* mode_0, counter_9 */
outportb(BASE+2,0xff); /* counter_9 low byte first */
outportb(BASE+2,0xff); /* counter_9 high byte ,0xffff=65535 */
}
/* ---------------------------------------------------------------------*/
void  interrupt irq5_service()
{
int i;
char c;

/* This program can process proper software protection */

watchdog=1;
outportb(0x20,0x20);
}
```

3.8 Pulse Width Measurement

- ***Clock1*** select 1.6M
- CH1 select ***clock1***=1.6M
- Pulse signal come into EXT_GATE1, pin 21 of CN1
- GATE1 select external gate control

- initial counter=0xffff
- If pulse signal=HIGH --> counter will down count
- Pulse width=(0xffff-current counter value+2) * Width of 1.6M CLK

```
/*-----------------------------------------------------------------------------*/
/* filename=demo_08.c                                                          */
/*                                                                             */
/* pulse width measurement                                                     */
/* IO BASE address = 0x200, all DIP switchs of SW1 in OFF position */
/*                                                                             */
/* CLOCK1=1.6M, CH1=CLOCK1=1.6M, GATE1=EXTERNAL GATE CONTROL
/* PULSE SIGNAL --> EXT_GATE1=PIN 21 OF CN1                                    */
/* CLK=1.6M=0.000625 ms                                                        */
/*-----------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>

#define  BASE 0x200            /* IO BASE address = 0x200 */

void main(void)
{
int i,high,low;
unsigned int width;

outportb(BASE+4,0);     /* select 8254 chip_1 --> counter_1,2,3 */
while ((inportb(BASE+5)&0x10)==0) ; /* wait until EXT_GATE1 HIGH */
while ((inportb(BASE+5)&0x10)!=0) ; /* wait until EXT_GATE1 LOW  */

/* CLK1=CLOCK1=1.6M */
outportb(BASE+3,0x30);       /* mode_0, counter_0 */
outportb(BASE,0xFF);   /* counter_0 low byte first */
outportb(BASE,0xFF);   /* counter_0 high byte */

while ((inportb(BASE+5)&0x10)==0) ; /* wait until EXT_GATE1 HIGH */
while ((inportb(BASE+5)&0x10)!=0) ; /* wait until EXT_GATE1 LOW  */

outportb(BASE+3,0x00);         /*latch counter_0 */
low=inportb(BASE)&0xff;
high=inportb(BASE)&0xff;
printf("\nhigh=%x,low=%x",high&0xff,low&0xff);
width=(0xff-high)*256+0xff-low+2;
printf("\nwidth=%u,high=%x,low=%x",width,high&0xff,low&0xff);
printf(" --> %f ms",(float)width*0.000625);
}
_
```