PCL-836

Multifunction counter-timer and digital I/O add-on card for PC/XT/ AT and compatibles

Copyright

Acknowledgments

Part No.2003836010 2nd Edition
Printed in Taiwan May 1997

Contents

CHAPTER 1

Introduction

1.1 Introduction

The PCL-836 is a multifunction counter-timer and digital I/O add-on card for IBM PC/XT/AT and compatibles. It provides six 16-bit down counters, a 10 MHz crystal oscillator timebase with divider and general purpose 16-bit TTL input and output ports. Four Intel 8254 (or compatible) counter/timer chips are used for all counting and timing functions.

1.2 Key Features

- 6 independent 16-bit counters
- 10 MHz maximum input frequency
- 10 MHz on-board timebase
- Binary or BCD counting
- Programmable frequency output
- Complex duty cycle outputs
- One-shot or continuous outputs
- 16-bit TTL digital input
- 16-bit TTL digital output
- Jumper selectable interrupt level

1.3 Applications

- Event counting for pulse output devices
- Programmable frequency synthesis
- Coincidence alarms
- Frequency measurements
- F/V conversion & pulse accumulation
- Period and pulse duration measurement
- Time delay measurement
- Periodic interrupt generation

1.4 Specifications

Programmable Counters:

- Counters: Six independent 16-bit counters
- Modes: Three programmable counter modes
- Programmable digital noise filter:

  1.6 µsec. to 52.428 msec.
- Usable pins: Clock, Gate and Out for each counter
- Programmable time-based output:

  153 Hz to 5 MHz
- 3 independent PWM outputs
- Input / Output TTL compatible
- Interrupt: IRQ 2,3,4,5,6,7,10,11,12,15 (Jumper select)

Digital Input / Output:

- 16 TTL input Channels

  Logical level 0 : 0.8 V max.

  Logical level 1 : 2.4 V min.
- 16 TTL output Channels

  Logical level 0 : 0.5 V max. @ 8.0 mA

  Logical level 1 : 2.4 V min. @ 0.4 mA

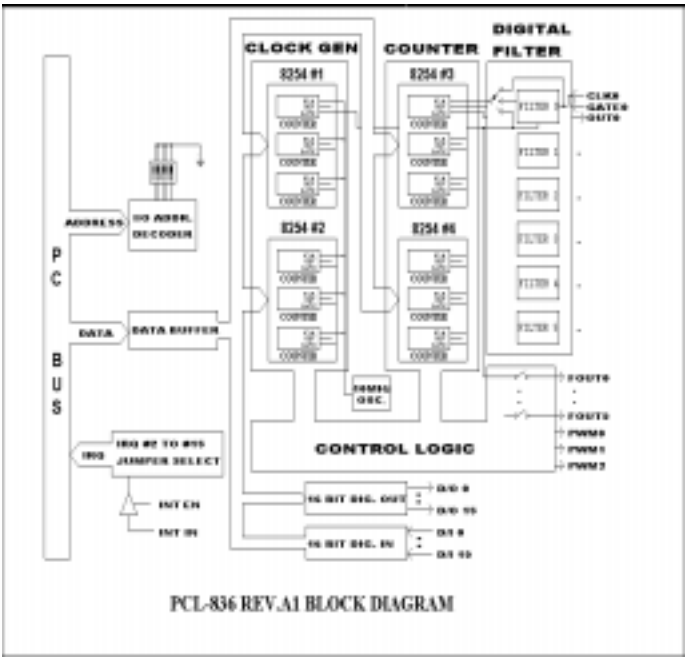
General:

- Connector:

  37 pin D-SUB connector for counter I/O

  20 pin male flat-cable connector for digital I/O
- Dimension: 185 mm x 100 mm
- The PCL-836 generates high frequency signals, which may cause EMI (Electromagnetic Interference) problems. Use of Advantech's shielded 37 pin D-SUB cable, or another well-shielded cable, avoids these problems.

1.5 Block Diagram



PCL-836 REV.A1 BLOCK DIAGRAM

CHAPTER 2

Installation

2.1 Initial Inspection

You should find the PCL-836 card and this user's manual inside the shipping container. The PCL-836 has been inspected and tested, both physically and electronically, before shipment. It should be free of marks and scratches and in perfect working order upon receipt.

Check the unit for any signs of shipping damage when unpacking. If there is any damage to the unit or if it fails to meet specifications, notify our service department or your local sales representative immediately. Call the carrier and retain the shipping carton and packing material for inspection by the carrier. We will arrange to repair or replace the unit.

Remove the PCL-836 interface card from its protective packaging. Keep the anti-vibration package. Whenever you are not using the card, store it in the package for protection.

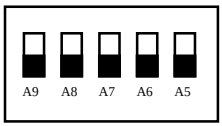
Discharge any static electricity that may have built up within your body by touching an unpainted surface on the back of your computer system before you handle the board. You should avoid contact with materials that create static electricity such as plastic, vinyl and Styrofoam. The board should be handled only by the edges to avoid static electric discharge that could damage the integrated circuits on the PCL-836.

2.2 Switch and Jumper Settings

There is one DIP switch (SW1) and one jumper (JP1) on the PCL-836 for the selection of the I/O address and interrupt level.

Switch name: SW1

The DIP switch SW1 is used to set base I/O address. Set the switch position to ON for logic 0 and to OFF for logic 1. The various base address settings are illustrated as follows:

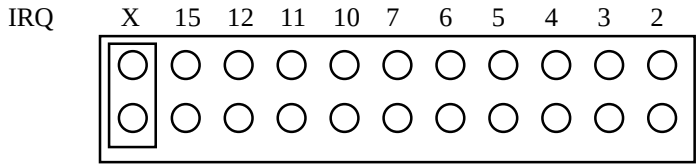


I/O address switch position					
(Hex)	1	2	3	4	5
	A9	A8	A7	A6	A5
200-1	0	0	0	0	
220-1	0	0	0	1	
240*	1	0	0	1	0
280-1	0	1	0	0	
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
3E0-1	1	1	1	1	

* Factory setting

Chose a base address that is not in use by another I/O device. A conflict with another device using the same I/O location will usually cause the PCL-836 and other devices to malfunction. The factory setting address is hex 240 which is usually kept free, as it is reserved for the PC prototype board.

Jumper name: JP1



The jumper JP1 is for selecting the interrupt level. Avoid using a level that is being used by another device.

Interrupt is enabled by setting the Interrupt Enable (CN3 pin 32) to logic low. The positive edge on the Interrupt input (CN3 pin 13) will then generate an interrupt if the 8259 interrupt controller on the system board is enabled. Using the interrupt implies that the user has installed an interrupt service routine and interrupt vectors to the service routine, and enabled the 8259 mask register for the level selected. It is not possible to set up an interrupt service routine using BASIC. It is usually necessary to use assembly language.

2.3 Connector Pin Assignments

There is one DB-37 connector and two 20-pin male connectors on the PCL-836. The connector CN3 is a counter I/O interface while the connector CN1 and CN2 are digital output and digital input respectively.

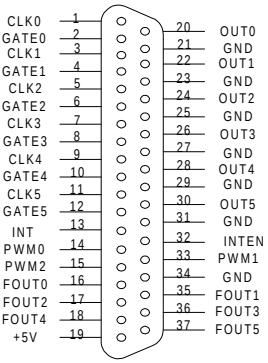
CN1	Digital output
CN2	Digital input
CN3	Counter signals and interrupt

The following diagrams illustrate the pin assignment of each connector:

Legend:

GATE	Gate of counter
CLK	Input of counter clock
OUT	Output of counter
PWM	PWM out
Fout	Frequency out
DO	Digital output
DI	Digital input
INTEN	Interrupt enable
INT	Interrupt input
GND	Ground

Connector CN3 - Counter signals and Interrupt Signals



PCL-836 REV.A1 D-TYPE
37PIN PIN ASSIGNMENT

Connector CN1 - Digital Output

DO0	1	2	DO1
DO2	3	4	DO3
DO4	5	6	DO5
DO6	7	8	DO7
DO8	9	10	DO9
DO10	11	12	DO11
DO12	13	14	DO13
DO14	15	16	DO15
GND	17	18	GND
+5V	19	20	+12V

CN1

Connector CN2 - Digital Input

DI0	1	2	DI1
DI2	3	4	DI3
DI4	5	6	DI5
DI6	7	8	DI7
DI8	9	10	DI9
DI10	11	12	DI11
DI12	13	14	DI13
DI14	15	16	DI15
GND	17	18	GND
+5V	19	20	+12V

CN2

2.4 Hardware Installation

Warning: TURN OFF your PC power supply whenever installing or removing the PCL-836 or connecting and disconnecting cables.

Installing the card in your computer:

1. Turn off the computer and all peripheral devices (such as printers and monitors).
2. Disconnect the power cord and any other cables from the back of the computer. Turn the system unit so the back of the unit faces you.
3. Remove the system unit cover (see your computer user's guide if necessary).
4. Locate the expansion slots at the rear of the unit and choose an unused slot.
5. Remove the screw that secures the expansion slot cover to the system unit. Save the screw to secure the interface card retaining bracket.
6. Carefully grasp the upper edge of the PCL-836 card. Align the hole in the retaining bracket with the hole on top of the expansion slot, and align the gold-striped edge connector with the expansion slot socket. Press the board firmly into the socket.
7. Replace the screw in the expansion slot retaining bracket.
8. Attach necessary accessories (DB-37 pin cable or connector adapter, etc.) to the interface card according to your application requirements.
9. Replace the system unit cover. Connect the cables you removed in step 2. Turn on the computer.

Hardware installation is now complete. Proceed to install the software driver.

CHAPTER 3

Register Structure and Format

3.1 I/O Port Address Map

The PCL-836 uses 32 consecutive addresses in the PC I/O address space. The base or start address (BASE) is selected by the DIP switch SW1. Each device on the PCL-836 has its own I/O location as follows:

PC/AT users should note that all ports are 8 bits (one byte) wide and should perform byte oriented read/write operations rather than word (16 bits) operations. When performing consecutive byte transfers to the same I/O port on the PC/AT, the PC AT Technical Reference Manual recommends the following coding in assembly language. This is required to allow sufficient recovery time for the AT I/O circuit:

```
        OUT IO_ADDR, AL     'write low byte
        JMP NEXT            'delay
NEXT:   MOV AL, AH          'fetch high byte
        OUT IO_ADDR, AL     'write high byte
```

PCL-836 I/O Port Address Map

I/O Address	Write	Read
	8254 chip 1	
BASE+0	Fout 0	
BASE+1	Fout 1	
BASE+2	Fout 2	
BASE+3	Control word	N/U
	8254 chip 2	
BASE+4	Fout 3	
BASE+5	Fout 4	
BASE+6	Fout 5	
BASE+7	Control word	N/U
	8254 chip 3	
BASE+8	Counter 0	
BASE+9	Counter 1	
BASE+10	Counter 2	
BASE+11	Control word	N/U
	8254 chip 4	
BASE+12	Counter 3	
BASE+13	Counter 4	
BASE+14	Counter 5	
BASE+15	Control word	N/U
	Digital input and output	
BASE+16	D/O low byte	D/I low byte
BASE+17	D/O high byte	D/I high byte
	Counter clock input mode	
BASE+18	Counter 0 control register	N/U
BASE+19	Counter 1 control register	N/U
BASE+20	Counter 2 control register	N/U
BASE+21	Counter 3 control register	N/U
BASE+22	Counter 4 control register	N/U
BASE+23	Counter 5 control register	N/U

Note: N/U = not used
8254 chip 1 and chip 2 are clock generators, 8254 chip 3 and chip 4 are counters

3.2 Counter control register format (BASE+18~23)

D7	D6	D5	D4	D3	D2	D1	D0	Counter mode
X	X	X	X	X	*	0	0	External clock without digital filter
X	X	X	X	X	*	0	1	External clock with digital filter
X	X	X	X	X	X	1	0	Internal clock
X	X	X	X	X	X	1	1	PWM mode

*D2 controls the external clock. D2=0 count the positive edge, D2=1 count the negative edge.

- **External clock without digital filter:** In this mode, the counter's clock input direct input from connector, and the maximum input frequency can be up to 10 MHz.
- **External clock with digital filter:** In this mode, the counter's clock input is passed through the digital filter, and the maximum input frequency depends upon the filter clock
- **Internal clock:** In this mode, the counter's clock input is connected to chip 1 or 2 Fout
- **PWM mode:** In this mode, the counter is configured for PWM function

Frequency output control register (BASE+24)

D7	D6	D5	D4	D3	D2	D1	D0
X	X	Fout 6	Fout 5	Fout 4	Fout 3	Fout 2	Fout 1

* Control the frequency output. Dn = 0 frequency output is off (3 state), Dn = 1. Frequency output is on.

3.3 Digital Input/Output

The PCL-836 suffers 16 bits of TTL compatible digital input and output. These digital input/output ports are at address BASE+16 and BASE+17. The data format of each port is as following:

Read Operation:

Base Address +16	D7	D6	D5	D4	D3	D2	D1	D0
D/I low byte	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

Base Address +17	D7	D6	D5	D4	D3	D2	D1	D0
D/I high byte	DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

Write Operation:

Base Address +16	D7	D6	D5	D4	D3	D2	D1	D0
D/O low byte	D07	D06	D05	D04	D03	D02	D01	DO0

Base Address +17	D7	D6	D5	D4	D3	D2	D1	D0
D/O high byte	DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

CHAPTER 4

Operation

The PCL-836 is a multifunction counter-timer and digital I/O add-on card for IBM PC/XT/AT and compatibles. There are four 8254 counter chips and 16 channels of TTL input/output on board, and each 8254 counter chip has 3 multifunction counters. Two counter chips (chip 1 and chip 2) are for the digital noise filter's sampling clock or frequency out, and the others (chip 3 and chip 4) are for the counters. For each counter clock input pin in chip 3 and chip 4, the PCL-836 features a Schmitt-trigger and a digital noise filter for noise immunity.

4.1 Counter modes

The PCL-836 has six multifunction counters. Users can program each counter in a different mode for their applications:

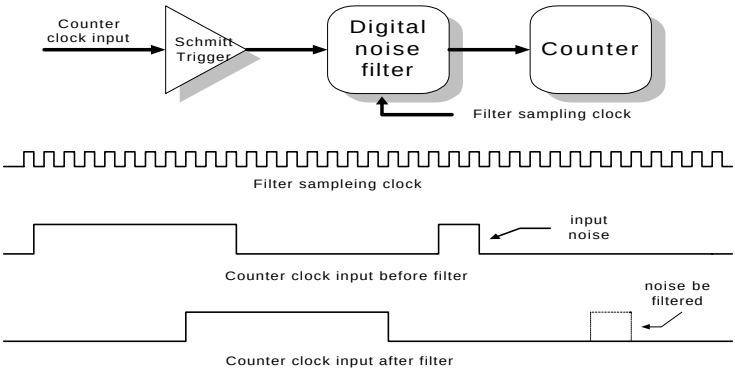
- Event counter
- Digital one-shot
- Programmable rate generator
- Square wave generator, etc.

For more information about programming the 8254 counter chip, refer to chapter 5 or consult 8254 chip product documentation. Counters 0, 1 and 2 are located in 454 chip 3 and counters 3, 4, and 5 are located in 8254 chip 4.

4.2 Digital noise filter

Noise immunity is the most important requirement for reliable counter operation. The PCL-836 conditions the clock input signals with a Schmitt-trigger and a programming digital filter. This filter reduces dips and spikes by sampling the clock input with a programmable filter sampling clock. The filter output waveforms change only when an input has the same value for seven consecutive sampling edges. The filter thus rejects noise or pulses shorter than seven sampling clock periods. You can optimize noise immunity by selecting a lower sampling frequency that is compatible with the highest input rate that you expect. For high speed clock input, users can disable the digital filter for their applications.

Each counter has its own noise filter. Users can program different filter sampling rates for different clock/event inputs. When the noise filter is enabled, the 8254's chip 1 and chip 2 have to set it in square wave mode to provide the digital noise filter's sampling clock.



Example: Enable the PCL-836 counter 0 digital noise filter

1. Write 01H to BASE+18H to enable counter 0 digital noise filter

2. Program counter 0 in 8254 operating mode 2

3. Program Fout 0 in 8254 operating mode 2

4.Set the Fout - frequency value

program in "C"

```
outportb(BASE+18, 0x01);   Enable counter 0 digital filter
outportb(BASE+11, 0x34);   Set counter 0 in 8254 operating mode2
outportb(BASE+03, 0x34);   Set Fout 0 in 8254 operating mode 2
outportb(BASE+00, 0x0A);   Set Fout - frequency value in low byte
outportb(BASE+00, 0x00);   Set Fout 0 frequency value in high byte
```

The filter sample clock (Fout) = 10 MHz/10 = 1 MHz

The filter sample clock is 1 MHz, and the period is 1ms, thus

1 x 7 = 7 (ms)

The minimum clock input high/low period must be > 7 ms.

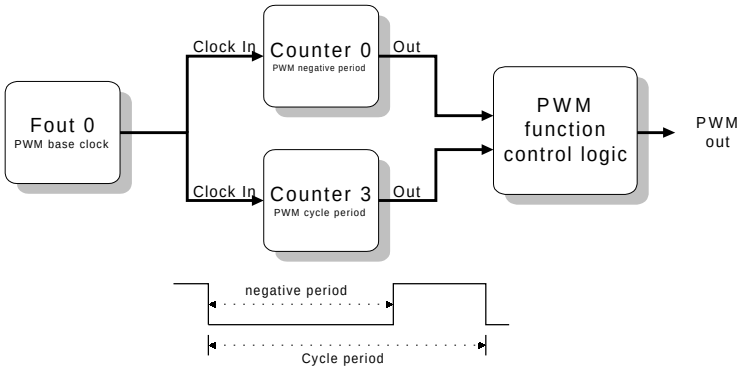
For more detailed information about how to program the digital noise filter, refer to the example program on the utility disk.

4.3 Pulse width modulation (PWM) function

The PWM function is very widely used in today's applications. The PCL-836 provides 3 PWM output channels.

When the PWM function is enabled, the PCL-836 will combine two counters as a PWM channel, and the counter in chip 3 must be programmed in mode 5, the counter in chip 4 must be programmed in mode 2 and the counter in chip 1 must be programmed in mode 3.

The PWM period and duty cycle are decided by the value we assign to the counters in chip 3, chip 4 and chip 1.




```
Example: Config PWM output 1
 1. Write the value 03H to BASE+18H, for set-
ting counter 0 in PWM mode
 2. Program the counter 0 in 8254 operating
mode 5
 3. Program counter 0 value for PWM negative
period
 4. Write the value 03H to BASE+21H, for set-
ting counter 3 in PWM mode
 5. Program the counter 3 in 8254 operating
mode 2
```

6. Program counter 3 value for PWM one cycle period

7. Program the Fout 0 in 8254 operating mode 2 for PWM base clock

8. Program the Fout 0 output frequency value

Program in C

```
outportb(BASE+18, 0x03);// Set counter 0 in
                        PWM mode
outportb(BASE+11, 0x3A); // Set counter 0 in
                        8254 operating mode 5
outportb(BASE+08, 0xFA); // Set low byte value
                        for PWM negative period
outportb(BASE+08, 0x00); // Set high byte
                        value for PWM negative
                        period
outportb(BASE+21, 0x03); // Set counter 3 in
                        PWM mode
outportb(BASE+15, 0x34); // Set counter 3 in
                        8254 operating mode 2
outportb(BASE+08, 0xE8); // Set low byte value
                        for PWM a cycle period
outportb(BASE+08, 0x03); // Set high byte
                        value for PWM a cycle peri-
                        od
outportb(BASE+03, 0x34); // Set Fout 0 in 8254
                        operating mode 2 for PWM
                        base clock
outportb(BASE+00, 0x0A); // Set Fout 0 fre-
                        quency value in low byte
outportb(BASE+00, 0x00); // Set Fout 0 fre-
                        quency value in high byte
```

The PWM base clock (Fout 0) = 10Mhz / 10 = 1Mhz

For instance, if the Fout 0 is programmed as a 1 mHz clock output, and the counter 0 value is 250, the counter 3 value is 1000, then:

The period of the PWM output will be 1000 X 1ms. = 1000 ms.

The negative period will be 250 X 1ms. =

CHAPTER 5

Programmable Timer/Counter

5.1 The Intel 8254

The PCL-836 uses 4 Intel 8254 programmable interval timer/counters. The 8254 is a very popular timer/counter device consisting of three independent 16-bit down counters. Each counter has a clock input, control gate and an output. It can be programmed to have a count from 2 up to 65535.

5.2 Counter Read/Write and Control Registers

The 8254 programmable interval timer uses four registers for each chip. The functions of each register are as follows:

BASE+0/4/8/12	Counter 0 Read/Write
BASE+1/5/9/13	Counter 1 Read/Write
BASE+2/6/10/14	Counter 2 Read/Write
BASE+3/7/11/15	Counter Control Word

Since the 8254 counter uses a 16-bit structure, each section of read/write data is split into the least significant byte (LSB) and the most significant byte (MSB). It is important to ensure that your read/write operations are in pairs and to keep track of the byte order.

The data format of the control register is:

BASE +	D7	D6	D5	D4	D3	D2	D1	D0
	SC1	SC0	RW1	RW0	M2	M1	M0	BCD

Legend:

SC1 & SC0: Select counter

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-back command

RW1 & RW0: Select the Read/Write operation

RW1	RW0	Operation
0	0	counter latch
0	1	Read/Write LSB
1	0	Read/Write MSB
1	1	Read/Write LSB first, then MSB

M2, M1 and M0: Select the operating mode

M2	M1	M0	Mode
0	0	0	0 interrupts terminal count
0	0	1	1 programmable one-shot
X	1	0	2 Rate generator
X	1	1	3 Square wave rate generator
1	0	0	4 Software triggered strobe
1	0	1	5 Hardware triggered strobe

BCD: Select binary or BCD counting

BCD	Type
0	Binary counter 16 bits
1	Binary coded decimal (BCD) counter

If the module is set to be binary, the count can be any number from 0 up to 65535. If the module is set to be BCD (binary coded decimal), the count can be set as any number from 0 to 9999.

If both SC1 and SC0 bits are set to 1, the counter control register is in read-back command. The data format of the control register then becomes:

BASE+	D7	D6	D5	D4	D3	D2	D1	D0
	1	1	CNT	STA	C2	C1	C0	X

Legend:

CNT = 0 latch count of selected counter(s)
STA = 0 latch status of selected counter(s)
C2, C1 and C0: Select counter for a read-back operation
C2 = 1 select counter 2
C1 = 1 select counter 1
C0 = 1 select counter 0

If SC1 and SC0 are both set to 1 and STA is set to 0, the counter read/write register selected by C2 to C0 contains a return status byte. The data format of the counter read/write register then becomes:

BASE+	D7	D6	D5	D4	D3	D2	D1	D0
	OUT	NC	RW1	RW0	M2	M1	M0	BCD

Legend:

OUT: Counter output current state

NC: Null count indicates when the last count written to the counter register has been loaded into the counting element.

5.3 Counter Operating Modes

5.3.1 Mode 0: Stop on terminal count

The output will initially be low after setting this mode of operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When the terminal count is reached, the output will go high and remain high until the selected counter is reloaded with the mode or a new count is loaded. The counter continues to decrement after terminal count has been reached. Rewriting a counter register during counting generates the following results:

1. Write the first byte stops the current counting

2. Writing the second byte starts the new count

5.3.2 Mode 1: Programmable One-Shot

The output will go low on the count following the rising edge of the gate input. The output will go high on the terminal count. If a new count value is loaded while the output is low, it will not affect the duration of the one-shot pulse until the following trigger. The current count can be read at any time without affecting the one-shot pulse. The one-shot is retriggerable, thus the output will remain low for the full count after any rising edge at the gate input.

5.3.3 Mode 2: Rate generator

The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the counter register. If the counter register is reloaded between output pulses, the present period will not be affected, but the subsequent period will reflect the value.

The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Therefore the gate input can be used to synchronize the counter.

When this mode is set, the output will remain high until the count register is loaded and the output can also be synchronized by software.

5.3.4 Mode 3: Square wave generator

Mode 3 is similar to mode 2, except that the output will remain high until one half of the count has been completed (for even numbers), and will go low for the other half of the count. This is accomplished by decreasing the counter by two on the falling edge of each clock pulse. When the counter reaches the terminal count, the state of the output is changed, the counter is reloaded with the full count and the whole process is repeated.

If the count is odd and the output is high, the first clock pulse (after the count is loaded) decrements the count by 1. Subsequent clock pulses decrement the count by 2. After time-out, the output goes low and the full count is reloaded. The first clock pulse (following the reload) decrements the counter by 3. Subsequent clock pulses decrement the count by two until time-out, then the whole process is repeated. In this way, if the count is odd, the output will be high for (N+1)/2 counts and low for (N-1)/2 counts.

5.3.5 Mode 4: Software triggered strobe

After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, and will then go high again.

If the count register is reloaded during counting, the new count will be loaded on the next CLK pulse. The count will be inhibited while the GATE input is low.

5.3.6 Mode 5: Hardware triggered strobe

The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable.

5.4 Counter operations

5.4.1 Read/Write Operation

For each counter, the type of read/write operation, operating mode and counter type must all be properly specified in the control byte, and the control byte must be written before the initial count is written.

Since the control byte register and all three counter read/write registers have separate addresses and each control byte specifies the counter that it applies to (through SC1 and SC0), no instructions on the operating sequence are required. Any programming sequence following the 8254 convention is acceptable.

There are three types of counter operation: read/load LSB, read/load MSB and read/load LSB followed by MSB. It is important to ensure your read/write operations are in pairs and to keep track of the byte order.

5.4.2 Counter read-back command

The 8254 counter read-back command allows users to check the count value, programmed mode and the current states of the OUT pin and Null Count flag of the selected counter(s). The command is written into the control word register and has the format shown in section 5.2.

The read-back command may be used to latch multiple counter output latches by setting the CNT bit = 0 and selecting the desired counter(s). The single command is functionally equivalent to several counter latch commands, one for each counter latched.

The read-back command can also be used to latch status information of selected counter(s) by setting the STA bit = 0. Status must be latched to be read; the status of a counter is accessed by a read from that counter. The counter status format is shown in section 5.2.

5.4.3 Counter latch operation

It is often desirable to read the value of a counter without disturbing the count in progress. Usually the method used is the counter latch command method which allows the user to read the latched count value of the selected counter.

The 8254 supports the counter latch operation in two ways. The first way is to set the RW1 and RW0 to be (0,0) which latches the count of the selected counter in a 16-bit hold register. The second approach is by performing a latch operation under the read-back command by setting the SC1 and SC0 to be (1,1) and CNT = 0. This method has the advantage of operating several counters at the same time. A subsequent read operation on the selected counter will retrieve the held value.

CHAPTER 6

PCL-836 Software Driver

The utility disk that came with your PCL-836 card includes some C library files. These libraries were developed using Turbo C, and you should be able to develop your own C applications using these files. The source code for the programming library can also be found on the floppy disk. This enables you to recompile the libraries using any C compiler, though some modifications may be necessary.

Function call descriptions

This section gives detailed descriptions of the functions available in the library files.

DeviceOpen

This function sets the device number and base address of the PCL-836. This enables the use of multiple PCL-836 cards

Prototype:
ULONG DeviceOpen(USHORT DeviceNumber, USHORT BaseAddress)

Parameters:

DeviceNumber	The board number of PCL-836, from 0 ~ 9
BaseAddress	The base address of PCL-836

Returned value:
0) success
1) Invalid base address
2) Invalid device number
3) Device is busy

Example:
```
error_code=DeviceOpen(3, 0x200)
```

note:

1. The user has to call the DeviceOpen function before the other functions.

2. After the I/O operations have been done, the user has to call DeviceClose to close this device.

DeviceClose

This function resets the PCL-836 to default status.

Prototype:
ULONG DeviceClose(USHORT DeviceNumber)

Parameters:
DeviceNumber The board number of the PCL-836, from 0~9

Returned value:
0) Success
1) Invalid device number
4) Lost base address

Example:
error_code=DeviceClose(3)

CounterConfig

This function configures the counter mode of a specified counter.

Prototype:
USHORT CounterConfig(
USHORT DeviceNumber,
USHORT Counter,
USHORT CounterEdge,
FLOAT MaxInFreq)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5
CounterEdge	The count edge, 0 for positive edge, 1 for negative edge
MaxInFreq	The maximum input frequency of the counter. The driver will automatically set the filter clock according to fMaxInFreq. If fMaxInFreq is set to zero, the driver will disable the filter function. The maximum frequency is 312 kHz

Returned value:
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter channel
8) Invalid frequency range

Example:
error_code=CounterConfig(3, 5, 0, 1000); Enables the filter function and the maximum input frequency is 1 kHz
error_code=CounterConfig(3, 5, 0, 0); Disables filter function

CounterEventStart

This function starts the counter counting.

Prototype:
ULONG CounterEventStart(
USHORT DeviceNumber,
USHORT Counter)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5

Returned value
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter channel

Example:
error_code = CounterEventStart(3, 5)

CounterEventRead

This function returns the counter value.

Prototype:
ULONG CounterEventRead(
USHORT DeviceNumber,
USHORT Counter,
USHORT *Overflow,
ULONG *Count)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5
Overflow	The status of counter overflow, 1 for 32-bit counter overflow, otherwise 0
Count	The counter value

Returned value:
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter channel

Example:
error_code = CounterEventRead(3, 5, &Overflow, &Count)

CounterFreqStart

This function configures the specified counter for frequency measurement, then starts the frequency measurement.

Prototype:

```
ULONG CounterFreqStart(
USHORT DeviceNumber,
USHORT Counter)
```

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5

Returned value:

0) Success
1) Invalid device number
4) Lost base address
5) Invalid channel number

Example:

```
error_code = CounterFreqStart(3, 5);
```

CounterFreqRead

This function reads the frequency measurement.

Prototype:
ULONG CounterFreqRead(
USHORT DeviceNumber,
USHORTCounter,
USHORT FreqLevel,
FLOAT *Freq)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5
FreqLevel (Hz)	=0 MAX 1Hz~65 kHz
	=1 MAX10 kHz~650 kHz
	=2 MAX100 kHz~6500 kHz
Freq	The frequency value

Returned value:
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter number

Example:
error_code = CounterFreqRead(3, 5, MAX1_65 kHz, & Freq)

CounterPulseStart

This function starts the pulse width modulation (PWM) output

Prototype:
ULONG CounterPulseStart(
USHORT DeviceNumber,
USHORT Counter,
float Period,
float UpCycle)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~2
Period	The total period in ms
UpCycle	The first 1/2 cycle length in ms

Returned value
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter channel
6) Invalid input parameter
8) Invalid frequency range

Example:
error_code = CounterPulseStart(3, 1, 0.5, 0.01)

CounterReset

This function resets the counter to power-on state.

Prototype:
ULONG CounterReset(
USHORT DeviceNumber,
USHORT Counter)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5

Returned value:
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter number

Example:
```
error_code = CounterReset(3, 5);
```

FreqOutStart

This function configures the specified counter for frequency output, then starts freqrency output.

Prototype:
ULONG FreqOutStart(
USHORT DeviceNumber,
USHORT Counter,
FLOAT Fout)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5
Fout	The frequency of output in Hz, from 153 Hz ~5MHz

Returned value:
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter number
8) Invalid frequency range

Example:
error_code = FreqOutStart (3, 5, 10000);

FreqOutReset

This function stops the frequency from the specified counter.

Prototype:
ULONG FreqOutReset(
USHORT DeviceNumber,
USHORT Counter)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Counter	The counter number of PCL-836, from 0~5

Returned value:
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter number

Example:
error_code = FreqOutReset(3, 5);

DioReadPortByte

This function reads the current digital input value from the specified digital I/O port.

Prototype:
ULONG DioReadPortByte(
USHORT DeviceNumber,
USHORT Port,
USHORT *Value)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Port	The port number of PCL-836, from 0~1
Value	The data of digital input

Returned value
0) Success
1) Invalid device number
4) Lost base address
9) Invalid port channel

Example:
error_code = DioReadPortByte(3, 0, &Value)

DioWritePortByte

This function writes digital output data to the specified digital port.

Prototype:
ULONG DioWritePortByte(
USHORT DeviceNumber,
USHORT Port,
USHORT Mask,
USHORT State)

Parameters:

DeviceNumber	The board number of PCL-836, from 0~9
Port	The port number of PCL-836, from 0~1
Mask	Specifies which bit(s) of data should be sent to the digital output port and which bits remain unchanged
State	New digital logic state

Returned value
0) Success
1) Invalid device number
4) Lost base address
5) Invalid counter number

Example:
error_code=DioWritePortByte(3, 0, 0x0F, State)

Note:
The previous state of the digital port should be stored with the configuration data.

DioReadBit

This function returns the bit state of digital input from the specified digital I/O port.

Prototype:
ULONG DioReadBit(
ULONG DeviceNumber,
USHORT Port,
USHORT Bit,
USHORT *State)

Parameters

DeviceNumber	The board number of PCL-836, from 0~9
Port	The port number of PCL-836, from 0~1
Bit	The bit number, from 0~7
State	The bit data read from the specified port, 0 or 1

Return value
0) Success
1) Invalid device number
4) Lost base address
6) Invalid Input parameter
9) Invalid port channel

Example:
error_code = DioReadBit(3, 0, 5, &State)

DioWriteBit

This function writes the bit state of digital output to the specified digital I/O port

Prototype:
ULONG DioWriteBit(
ULONG DeviceNumber,
USHORT Port,
USHORT Bit,
USHORT State)

Parameters

DeviceNumber	The board number of PCL-836, from 0~9
Port	The port number of PCL-836, from 0~1
Bit	The bit number, from 0~7
State	The bit data read from the specified port, 0 or 1

Returned value
0) Success
1) Invalid device number
4) Lost base address
6) Invalid input parameter
9) Invalid port channel

Example:
error_code = DioWriteBit(3, 0, 5, 1)

Note:
The previous state of the digital port should be stored with the configuration data.

DioGetCurrentDOByte

This function returns the digital output data from the specified digital I/O port.

Prototype:
ULONG DioGetCurrentDOByte(
USHORT DeviceNumber,
USHORT Port,
USHORT *Value)

Parameters:

DeviceNumber	Board number of PCL-836, from 0~9
Port	Port number of PCL-836, from 0~1
Value	8-bit digital data of specified output port

Returned value:
0) Success
1) Invalid device number
4) Lost base address
6) Invalid input parameter
9) Invalid port channel

Example:
error_code = DioGetCurrentDOByte(3, 0, &Value)

DioGetCurrentDOBit

This function returns digital output data from the specified digital I/O port

Prototype:
ULONG DioGetCurrentDOBit(
USHORT DeviceNumber,
USHORT Port,
USHORT Bit,
USHORT *State)

Parameters

DeviceNumber	Board number of PCL-836, from 0~9
Port	Port number of PCL-836, from 0~1
Bit	8-bit digital data of specified output port, 0~7
State	Bit data from the specified port

Returned value:
0) Success
1) Invalid device number
4) Lost base address
6) Invalid input parameter
9) Invalid port channel

Example:
error_code = DioGetCurrentDOBit(3, 0, 5, &State)