PCL-839
3-axis High Speed
Stepping Motor Control Card

PC-LabCard Series
User's Manual

Copyright

This documentation and the software routines contained in the PCL-839 software disk are copyrighted 1994 by Advantech Co., Ltd. All rights are reserved. Advantech Co., Ltd. reserves the right to make improvements in the products described in this manual at any time without notice.

No part of this manual may be reproduced, copied, translated or transmitted in any form or by any means without the prior written permission of Advantech Co., Ltd. Information provided in this manual is intended to be accurate and reliable. However, Advantech Co., Ltd. assumes no responsibility for its use, nor for any infringements of the rights of third parties which may result from its use.

Acknowledgments

PC-LabCard is a trademark of Advantech Co., Ltd. IBM and PC are trademarks of International Business Machines Corporation. MS-DOS, Microsoft C and Quick Basic are trademarks of Microsoft Corporation. BASIC is a trademark of Dartmouth College. Intel is a trademark of Intel Corporation. Turbo C is a trademark of Borland International.

Contents

Finding your way around in this manual

This manual is organized in five chapters, and contains three appendixes with additional information. The information contained in each chapter is as follows:

Chapter 1: General Information
If you have just purchased the PCL-839, or just need to brush up on its Features/specifications, you would want to read this chapter.

Chapter 2: Installation
If you have not yet configured and/or installed your PCL-839, or need to change the configuration (e.g. set a different base address), this chapter will give you the information you require.

Chapter 3: PROG839CommandInterpreter
This chapter describes the utility command interpreter included with the PCL-839. If you want to program the PCL-839, and have not yet used the command-set of the PCL-839, this utility program is an excellent tool to help you learn. Chapter 5 describes the PCL-839's hardware registers. If you are not familiar with these registers, it is advisable to read Chapter 5 prior to starting with this chapter.

Chapter 4: PCL-839 Software Library
This chapter describes the 'C' libraries and the functions they contain. If you want to write your own applications in 'C', this chapter will give you all the information you need. If you are not familiar with the hardware registers (and the naming conventions) of the PCL-839, read Chapter 5 before continuing with this chapter.

Chapter 5: Register Programming
This chapter describes the PCL-839's hardware registers. It also contains typical operational procedures that will assist you in program design. This chapter is a good place to start getting to know and use the capabilities of the PCL-839 to best suit your application.

1
General information

Introduction

The PCL-839 is a high-speed three-axis stepping motor control card that simplifies stepping motor control, giving you added performance from your stepping motors.

Three-axis control

The PCL-839 has three single-chip pulse generators on-board, which enables the simultaneous and independent control of three axis. The PCL-839 provides digital pulse and directional control (+ and -) for each stepping motor axis.

User-friendly interface

The PCL-839 has been designed to act as a user-friendly solution for your stepping motor control applications. Programming the PCL-839 is very easy. 'C' Libraries are provided and they contain all the command functions needed for total control of your stepping motors.

Stand-alone interpreter

A stand-alone, non-resident command interpreter, PROGg39.EXE, can also be used to control your stepping motors without any programming.

Digital I/O

The PCL-839 features 16 digital inputs and 16 digital outputs for general I/O use (on/off control etc.).

Isolation protection

The PCL-839's PULSE and DIRECTION outputs and five limits input switches are isolated from the PC side.

Features

- Three on-board pulse generators that enables simultaneous independent control of three stepping motors

- Two operating modes - two-pulse (+ and - direction pulse) or one-pulse (pulse-direction) mode

- Programmable step rate from 1 to 16k pps (pulses per second).

- Programmable initial speed, final speed and time duration. Automatic trapezoidal acceleration/deceleration Tamping is performed

- 16 I/O TTL compatible channels

- All inputs/outputs are optically isolated, providing 500VDC isolation protection

- 'C' libraries containing device drivers provided

- Command Interpreter provided that eases learning the PCL-839 command set

Applications

- Precise X-Y-Z position control

- Precise rotation control

- Robotics and assembly equipment

- Other stepping-motor applications

Specifications

- **No. of Axes**: Three independent axes (individually programmable)

- **Operating modes**: Two-pulse mode (+ or - direction) or one-pulse (pulse-direction) mode

- **Steps per command**: ±262,143 in normal mode, ±524,286 in double-frequency mode.

- **Step Rate**: 1 - 8k pps in normal mode, 1 - 16k pps in double-frequency mode

- **Acceleration/deceleration ramping**: User programmable start, run and ramping rates.

- **Output polarity**: Positive/negative going pulse (programmable)

- **Pull-up voltage**: external +5 V - +12 V

- **Output protection**: Opto-coupled with 1 kΩ pull-up resistor

- **Output driving capacity**: 20 mA @ 0.4 V_{DC}

- **Limit switch inputs**: 2 "Emergency stop" inputs, 2 "Slow down/ Accelerate" inputs and 1 "ORG" input. All limit switches are isolated from the PC.

- **Limit switch input voltage**: external +5 V to +12 V

- **Interrupt channels**: IRQ 2, 4, 5, 7, 10, 11, 12 or 15 (jumper selectable)

- **Limit switch types**: Normal-open (NO) or normal-closed (NC) - jumper selectable

Digital input/output

- **No. of input channels**: sixteen (+5V TTL compatible)

- **No. of output channels**: sixteen (+5V TTL compatible)

- **I/O address range**: sixteen consecutive I/O addresses

General

· Power consumption: 300mA, +5VDC

· Connector: 37-pin D-type connector

· Board dimensions: 183.5 mm x 99.06 mm

· Operatingtemperature:0 to70°C

Block diagram

PCL-839 HIGH SPEED STEPPING MOTOR CONTROL CARD
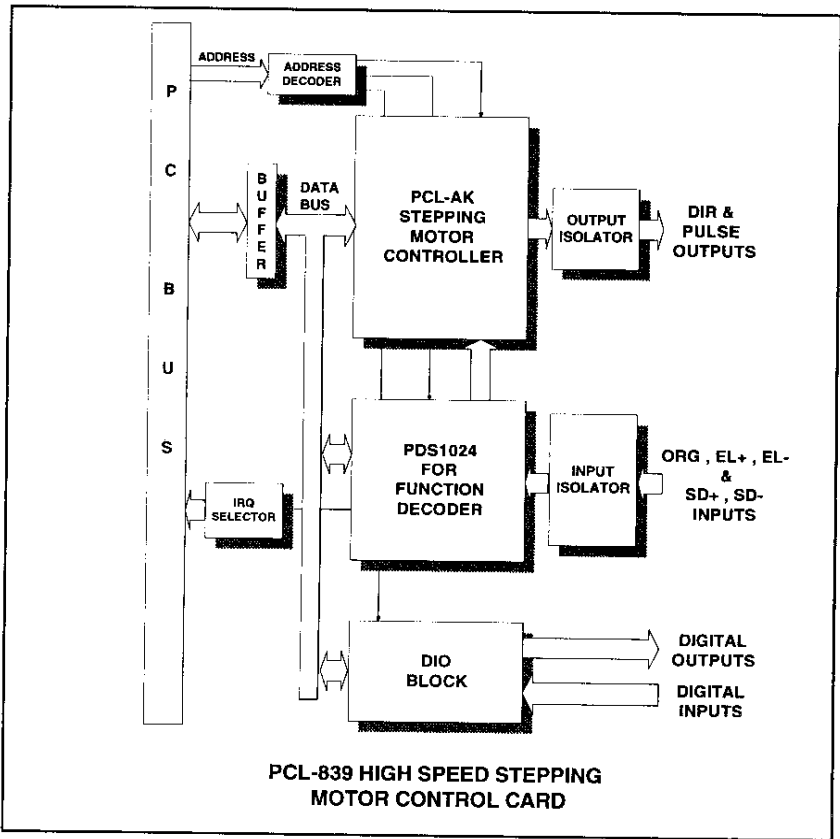
2
Installation

Switch and jumper settings

Before you install the PCL-839, you need to select the card's base address, set the limit switch configurations and the interrupt level that the card will use.

This section describes this procedure in detail.

Setting the PCL-B39 Base I/O address (S1)

The PCL-839 requires 16 consecutive I/O addresses. DIP switch S1 (shown below) sets the base I/O address.

Choose a base address that is not in use by any other I/O device. A conflict with another device may cause one or both devices to fail. The factory address setting (hex 300) is usually free as it is reserved for PC prototype boards.

Switch settings for various base addresses appear below:

Card I/O addresses (S1)

Range (hex)	Switch position						
	9	8	7	6	5	4	<=Addr. Line
200 - 20F	●	O	O	O	O	O	
210 - 21F	●	O	O	O	O	●	
⋮							
* 300 - 30F	●	●	O	O	O	O	
⋮							
3F0 - 3FF	●	●	●	●	●	●	

O = On   ● = Off   * = default

Note: Switches 1-6 control the PC bus address lines as follows:

Switch	1	2	3	4	5	6
Line	A9	A8	A7	A6	A5	A4

ON
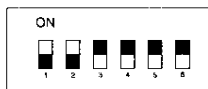
S1 - Base Address Settings

Limit Switch Configuration (JP1, JP2, JP3)

The PCL-839 features 5 limit switches for additional control of the output.

EL+/ EL-

These are the End Limit signal inputs. When the signal of the same direction as the pulse output (in direction or pulse mode) becomes active, pulse output stops immediately.

SD+/SD-

These are the Slow-Down signal inputs. They are in operation in the SD-enable mode (refer to the control select modes). When the signal of the same direction as the pulse output (in direction or pulse mode) becomes active during high-speed start, the frequency ramps down.
When the signal becomes in-active, the frequency ramps up again.

ORG

This is the Origin point input. When this signal becomes active during origin return (refer to the control select modes), pulse output stops immediately.

Although the PCL-839 caters for five limit switches, not all of them have to be operation in one application. Refer to Fig 2-1 (on the next page) for an example of the use of limit switches.

Limit Switch Polarity Setting

JP1, JP2 and JP3 set the polarity for channels C, B and A respectively.
When the jumper is set to LO (normal), the limit switch uses 'normally open' as default . When the jumper is set to HI, the limit switch uses' normally closed' as default.

JP1, JP2 and JPJ selection

    HI   Normally Closed
    LO   Normally Open

The figures on the next page illustrate limit switch use and settings.
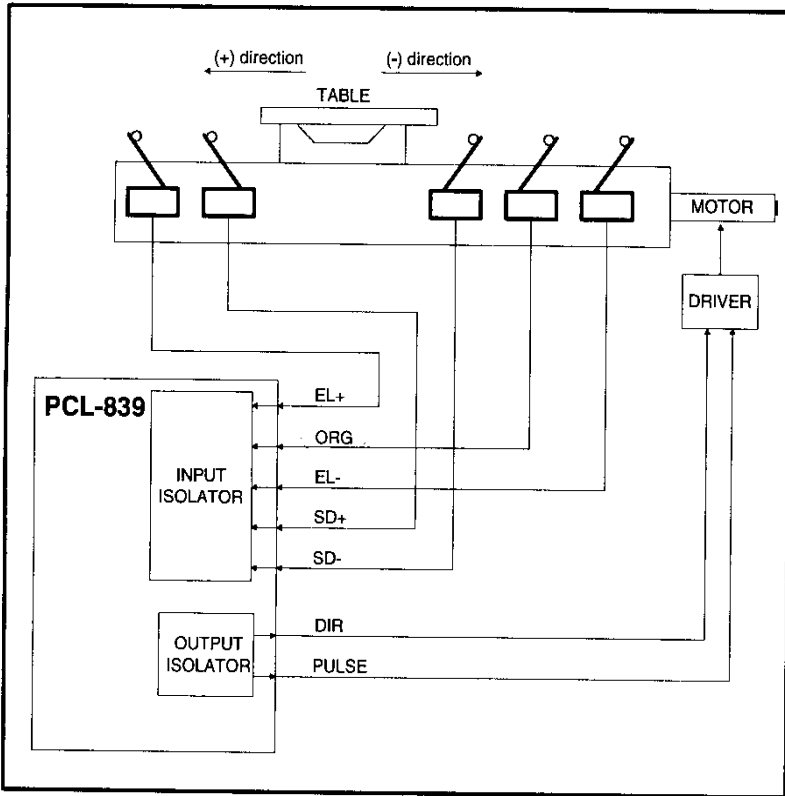
Using Limit Switches
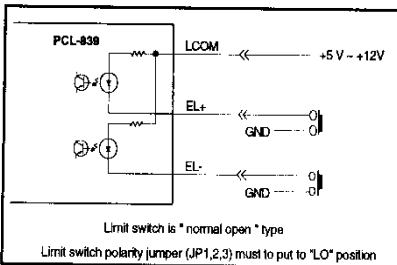
Normally-open wiring
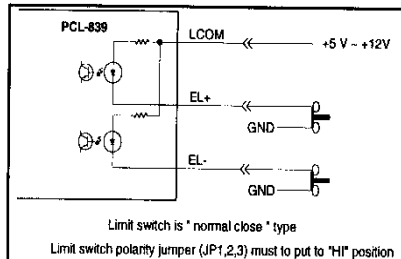
Normally-closed wiring

Interrupt level selection (JP4)

You have to set jumper JP4 to select the card's interrupt level (2, 4, 5, 7, 10, 11, 12 or 15), as shown below:

Card interrupt (default = 7)

```
      IRQ  15 12 11 10  7  5  4  2
           ┌ o  o  o  o ┌o┐ o  o  o
      JP4  └ o  o  o  o └o┘ o  o  o
```
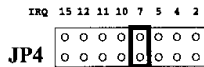
Do not select a level that is being used by another device unless you have performed special programming to share several devices on one interrupt. You can also control interrupt generation by software. If the interrupt is enabled by software, and the PCL-839 completes a motion, it will generate an interrupt. Your program can then determine which channel caused the interrupt by reading the status register.

Hardware Installation

After you have set the base address, limit-switch configuration and the interrupt level (as described in the previous section), you will be ready to install the card in your PC's chassis. The following section will assist you in installing the PCL-839.

Warning! *Disconnect power from your PC whenever you install or remove the PCL-839 or its 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 chassis so that the back of the unit faces you.

3. Remove the chassis cover (see your computer users 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 chassis. Save the screw to secure the PCL-839.

6. Carefully grasp the upper edge of the PCL-839 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. Replace the chassis cover.

9. Connect the D-37 male connector to the PCL-839's 37-pin female connector Connect the connector to your stepping motor driver according to the specifications outlined in Section 3.1.

10. Connect the cables you removed in step 2. Turn on the computer.

Hardware installation is now complete.

PCL-839 Pin Connections

This section assists you in connecting the PCL-839's 37-pin connector (located at CN3) to a variety of stepping motor drivers.

The following diagrams give the PCL-839's pin connector assignments, and offer some examples of input/output circuit connections from the card to the driver. You should select the example that best supports your application needs and the capabilities of your stepping motor driver.

Note: *Output circuit diagrams of stepping motor can be found in Appendix A.*

The 37-pin female connector (CN3)

```
DIR/-dir    ( CHA )      1  ○   ○ 20    PULSE/+dir  ( CHA )
EXT.VCC     ( CHA )      2  ○   ○ 21    COM          ( CHA )
NC                       3  ○   ○ 22    NC
DIR/-dir    ( CHB )      4  ○   ○ 23    PULSE/+dir  ( CHB )
EXT.VCC     ( CHB )      5  ○   ○ 24    COM          ( CHB )
NC                       6  ○   ○ 25    NC
DIR/-dir    ( CHC )      7  ○   ○ 26    PULSE/+dir  ( CHC )
EXT.VCC     ( CHC )      8  ○   ○ 27    COM          ( CHC )
NC                       9  ○   ○ 28    EL+          ( CHA)
EL-         ( CHA )     10  ○   ○ 29    SD+          ( CHA )
SD-         ( CHA )     11  ○   ○ 30    ORG          ( CHA )
LCOM        ( CHA )     12  ○   ○ 31    NC
EL+         ( CHB )     13  ○   ○ 32    EL-          ( CHB )
SD+         ( CHB )     14  ○   ○ 33    SD-          ( CHB )
ORG         ( CHB )     15  ○   ○ 34    LCOM         ( CHB )
NC                      16  ○   ○ 35    EL+          ( CHC )
EL-         ( CHC )     17  ○   ○ 36    SD+          ( CHC )
SD-         ( CHC )     18  ○   ○ 37    ORG          ( CHC )
LCOM        ( CHC )     19  ○
```
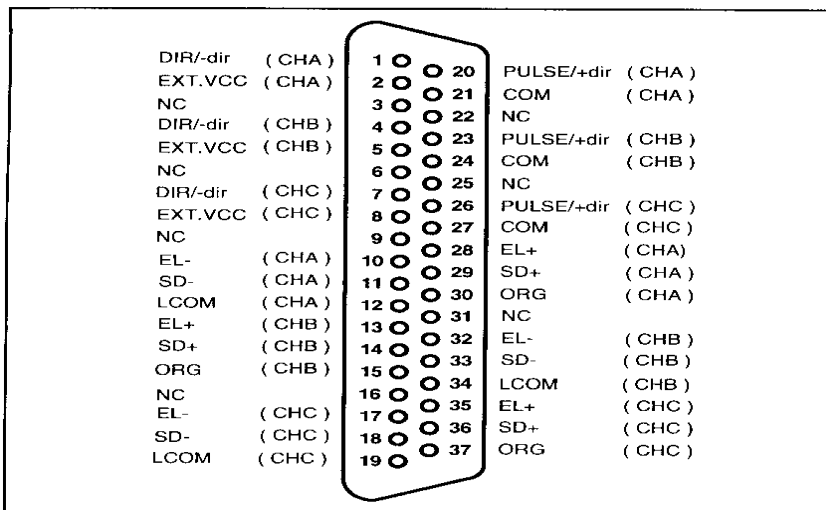
DIR/-dir :   direction signal output(in direction mode) or
                (-) direction pulse output (in pulse mode)

PULSE/+dir: pulse signal output(in direction mode) or
                (+)direction pulse output (in pulse mode)

EXTVCC: external power input

COM :   isolated outputs common point for each channel

EL+ :   (+)direction emergency stop limit switch input

EL- :   (-)direction emergency stop limit switch input

SD+ :   (+)direction slowdown limit switch input

SD- :   (-)direction slowdown limit switch input

ORG :   original (home) point limit switch input

LCOM :   limit switch common point for each channel

Example input/output circuit connections

The figure below illustrates an isolated output connection from the PCL-839 to the stepping motor driver
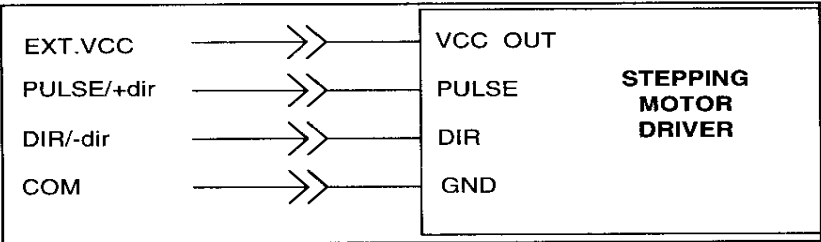
Isolated output connection

The next figure illustrates a non isolated connection where the PC's +12 V output bias is used.
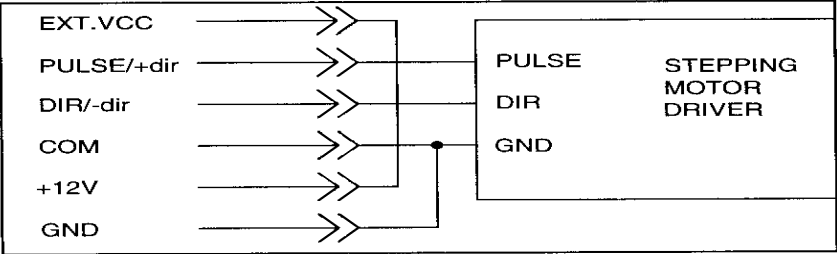
Non-isolated +12 V bias connection

The next two figures illustrate a TTL compatible output circuit connection and a current-drive output connection between the PCL-839 and the stepping motor driver.

TTL compatible output connection
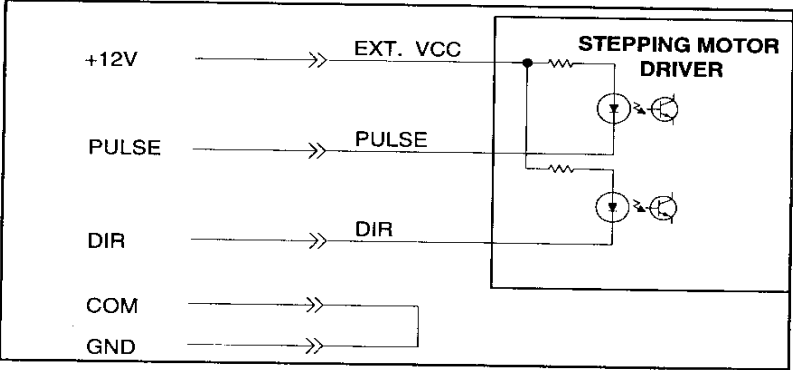
Current drive output connection

Digital Input and Output Connectors (CN1,CN2)

The PCL-839 provides two 20-pin digital input and output connectors, located at CNI (digital output) and CN2 (digital input). A variety of daugtherboards can be connected to these connectors. The PCLD-782B Isolated D/I board, the PCLD-785B Relay Output Board and the PCLD-786 SSR & Relay Driver Board are just three examples. The pin assignments for these connectors are given below.
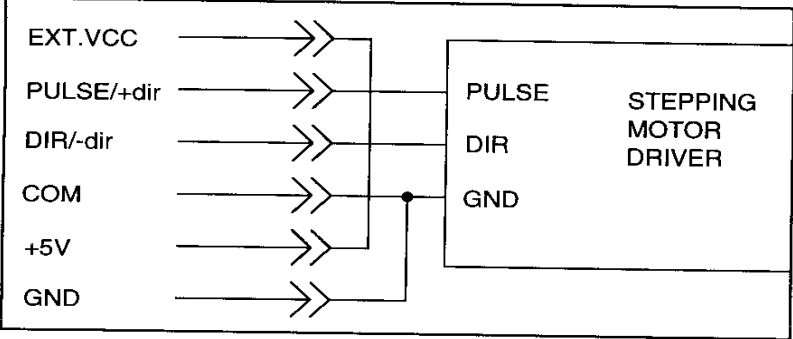
```
        D/O  0    1    2    D/O  1
        D/O  2    3    4    D/O  3
        D/O  4    5    6    D/O  5
        D/O  6    7    8    D/O  7
        D/O  8    9   10    D/O  9
        D/O 10   11   12    D/O 11
        D/O 12   13   14    D/O 13
        D/O 14   15   16    D/O 15
        GND      17   18    GND
        +5V      19   20    +12V
              CN1 ( DIGITAL OUTPUT )
```

CN1 (D/O) pin assignments

```
        D/I  0    1    2    D/I  1
        D/I  2    3    4    D/I  3
        D/I  4    5    6    D/I  5
        D/I  6    7    8    D/I  7
        D/I  8    9   10    D/I  9
        D/I 10   11   12    D/I 11
        D/I 12   13   14    D/I 13
        D/I 14   15   16    D/I 15
        GND      17   18    GND
        +5V      19   20    +12V
              CN2 ( DIGITAL INPUT )
```

CN2 (D/I) pin assignments

3
Prog839 Command Interpreter

Introduction

Included with the PCL-839 card is a utility command-interpreter, PROG839.EXE. PROG839.EXE is a convenient learning tool for familiarizing yourself with the command set of the PCL-839, and the PCL-839 itself. This chapter describes the commands supported by PROG839.EXE.

It is not recommended for use with stepping motor applications-software libraries that have been included for that purpose. When you write applications for your PCL-839, you can either use the included 'C' libraries or directly access the registers of the PCL-839 (as described in Chapter 4).

Using PROC839.EXE with textfiles.

PROG839.EXE is driven by commands contained in a ASCII textfile. You have to create this textfile using any ASCII-capable editor (or use the DOS *edit* command).

To execute PROG839.EXE, use the following DOS command :

```
PROG839 [filename] [ENTER],
```

where [filename] is the name of a ASCII textfile that contains the commands that will be interpreted by PROGg39.EXE (and drive the PCL-839).

For example, to interpret commands from the ASCII textfile "CURVE.TXT", you would enter

```
PROG839 CURVE.TXT [ENTER]
```

Once you have entered this command line, PROG839.EXE will display the commands from the textfile on you computer's monitor and will run the stepping motor.

When PROG839.EXE is busy executing the 'script', you can abort this process by pressing [ESC]. PROG839.EXE will complete the current step and then exit.

PCL-839aCommand Set

List of commands

Immediate commands

Command & *Parameters*   **Command Description**

/* This is a Comment */

✎ $ARC\ (CH\#_1,\ CH\#_2), dira(X_1, Y_1, X_2, Y_2)$

Use channels $CH\#_1$ & $CH\#_2$ to draw an arc from point (X_1, Y_1) to (X_2, Y_2), in direction *dira*

BASE *(port address)*

Set PCL-839 Base Address

✎ $CIRCLE\ (CH\#_1, CH\#_2), dirc(r)$

Use channels $CH\#_1$ & $CH\#_2$ to draw a circle with radius *r* steps, in direction *dirc*

ECHO *"string"*

Display a string on the screen

DEBUG *ON*

Activate DEBUG mode

DEBUG *OFF*

Deactivate DEBUG mode (default)

DISPLAY *ON*

Display the commands on the screen

DISPLAY *OFF*

Deactivate "DISPLAY ON"

IN *port #*

Read the value of an input port and display it on the screen

✎   *These commands are useful when you are using two stepper motors in a plotter configuration.*

✍ `LINE (CH#`$_1$`,CH#`$_2$`),(X`$_L$`,Y`$_L$`)`

Use channels CH#$_1$ & CH#$_2$ do draw a line from the current position to (X_{steps}, Y_{steps}) position

`LOOP (count#)`

Loop "count#" times

`LOOPEND`

End of Loop

`MANUAL`

Manual operation to move or adjust each axis parameter

`OUT port#, value`

Output value *'value'* to digital output port *'port#'*

`RUN`

Execute delayed commands

`SET (CH#[,CH#[,CH#]]), (FL,FH,AD)`

Set CH# speed

`SETMODE (CH#[,CH#[,CH#]]),mode`

Set output mode of *CH#* to "DIRECTION" or "PULSE" mode

`SLOWDOWN (CH#[,CH#[,CH#]])`

Slow *CH#* down to FL speed

`SLSTOP (CH#[,CH#[,CH#]])`

Slow *CH#* down to FL speed and then stop

`STOP (CH#[,CH#[,CH#]])`

Stop *CH#* immediately, also resets *CH#*

`STATUS`

Display the current status of all CH#

`WAITDI port#, value`

Wait until *port#*'s D/I = *value*

`WAITKEY`

Wait until any key is pressed

WAITRDY *(CH#[,CH#[,CH#]])*
  Wait until *CH#* is ready

WAITTIME *(milliseconds)*
  Delay for *milliseconds* ms.

Delayed commands (executed by the RUN command)

LSPORG dir*(CH#[,CH#[,CH#]])*
  Low speed move until limit switch ORG is ON

HSPORG dir*(CH#[,CH#[,CH#]])*
  High-speed move until"ORG" is ON.

LSPCMOVE dir*(CH#[,CH#[,CH#]])*
  Low speed move until "STOP" is executed.

HSPCMOVE dir*(CH#[,CH#[,CH#]])*
  High speed move until "SLOWDOWN" or"SLSTOP" is executed.

LSPPMOVE dir*(CH#[,CH#[,CH#]]),step#*
  Low speed move with preset value *step#*

HSPPMOVE dir*(CH#[,CH#[,CH#]]),step#*
  High speed move with preset value *step#*

Command descriptions

This section contains information on how to use the PCL-839 command set. A brief description, as well as the correct syntax, is given for every command. Examples are also provided for some commands.

PROG839.EXE has two kind of commands: immediate commands and delayed commands. When PROG839.EXE interprets an immediate command, it executes it immediately. When a delayed command is interpreted: it is stored and only executed when the next 'RUN' command is interpreted. Only one delayed command can be stored for each channel, and if two or more delayed commands (for the same channel) is found, only the last command will be executed when the 'RUN' command is interpreted. PROG839.EXE is not case-sensitive, therefore commands in upper-, lower- or mixed- case will be executed similarly.

The comment string

/* This is a comment */

The above format is used to leave comments in your textfile. PROG839.EXE will not interpret these lines, nor will it display them on the screen.

ARC (CH#1,CH#2,), dira (X1.Y1,.X2,,Y2,)

This command is used to draw an are using two stepping motors.

Format:
```
ARC     (CH#1,CH#2,), dira (X1.Y1,X2.Y2)
        CH#1.2= Channel numbers
        dira = direction (CW or CCW)
        X1,Y1, = starting coordinates of the are
        X2,Y2, = final coordinates of the are
```
Example:
```
ARC (1,2),CW(20,5,5,20)
```
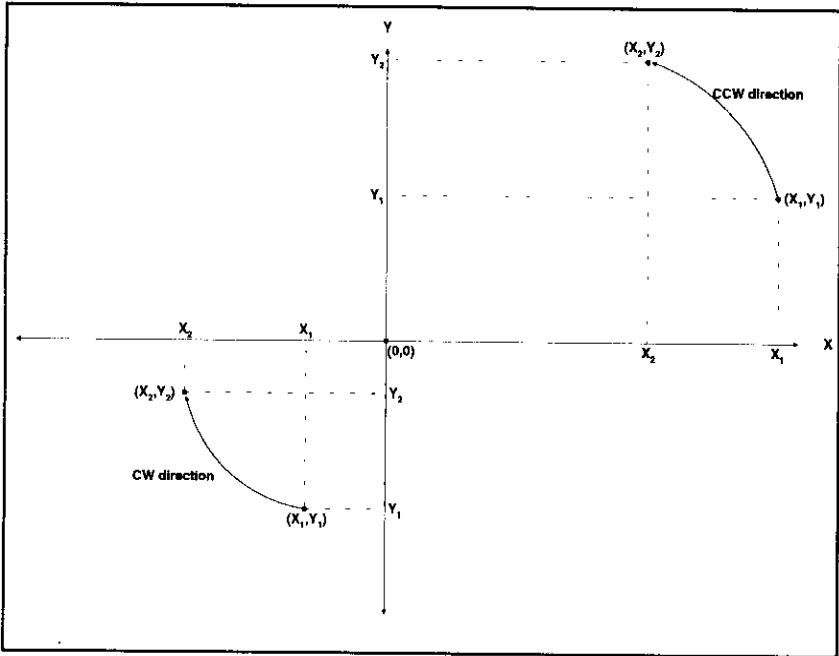
Using the ARC command

BASE (port address)

Sets the PCL-839's base I/O address. This enables you to control multiple PCL-839 cards form one PC, should it be required.

Format:

    BASE (port_address)
        port_address = base address

Erample:

BASE (0x2CO) /* Sets the base addr to 2C0 Hex. */
BASE (704)   /* Sets the base addr to 704 Decimal */
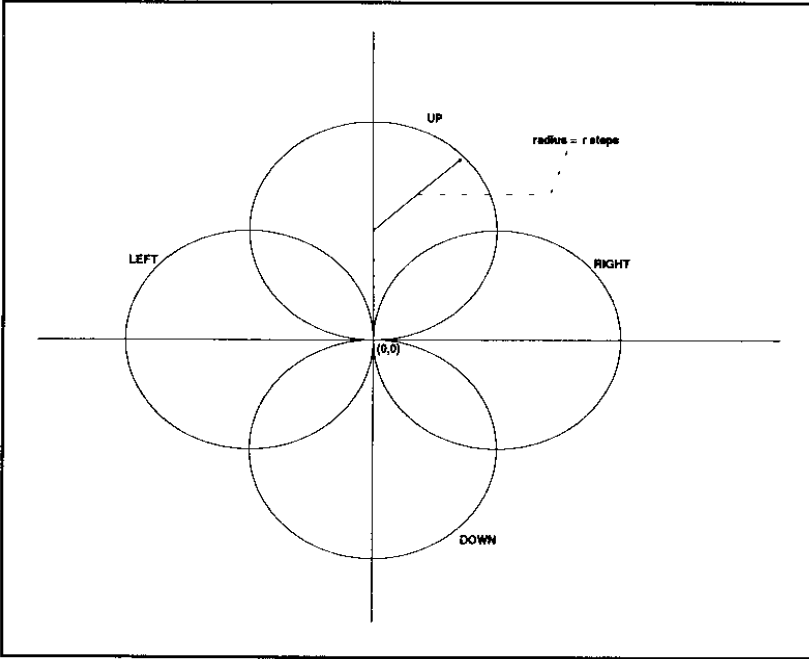
Using the Circle command

CIRCLE (CH#$_1$,CH#$_2$), dirc(r)

Uses two channels to draw a circle (like a plotter).

Format:

```
CIRCLE (CH#₁,CH#₂), dirc (r)
```
CH#$_{1,2}$ are the channels
dirc = Direction (UP, DOWN, LEFT, RIGHT)
r = radius of circle (in steps)

Example(s) :

```
CIRCLE (2, 3), UP (250) /* draws a circle "UP", with
                          radius 250 steps */
```

ECHO "string"

Displays a line of text on the screen.

Format:

```
ECHO "string"
string = any line of text
```

Example(s) :

```
ECHO "This line will be displayed "
ECHO "The stepper motor is running... "
```

DEBUG ON

Switches PROG839.EXE to debug mode. In this mode, PROG839.EXE will display the commands on the screen, but will not output anything to the stepping motor driver. This enables the user to check the syntax of the textfile, and the channel parameters.

Format:
DEBUG ON

There are 10 commands that will not affect the output in debug mode. They are/* comment */, BASE, MANUAL, CLR, ECHO,WAITKEY, WAITTIME, LOOP, LOOPEND, STATUS and DEBUG OFF.

DEBUG OFF (default setting)

Reverses DEBUG ON (see above). All commands will be interpreted and run after this command is interpreted.

DISPLAY ON (default setting)

Tells PROG839.EXE to display each command and its parameters on the screen when it is being interpreted.

Format:

```
DISPLAY ON
```

DISPLAY OFF

This inhibits display from PROG839.EXE. There are 4 commands that will not be affected by this command. They are ECHO, MANU AL, WAITKEY and WAITDI. Error messages from the PCL-839 will also still be displayed.

Format:

```
DISPLAY OFF
```

Example:

```
DISPLAY OFF
BASE (0x2C0)
DISPLAY ON
```

The above commands will set the base address to 2C0 (hex) but will not display anything on the screen.

IN port

The PCL-839 has two digital input ports. One of these ports are read, and the value displayed on the screen.

Format:

```
IN port#
```
   Port # is 0 or l

Example:

```
IN 0
```

Digital input port O is read, and the value is displayed on the screen.

LINE (CH#1, CH#2), (XL, YL)

Purpose:

This command draws a line using **2** stepping motor channels. The coordinates used are relative to the current position.

Format:

LINE (CH#1.CH#2). (XL,YL)

   CH#1,2 are the channels to be used
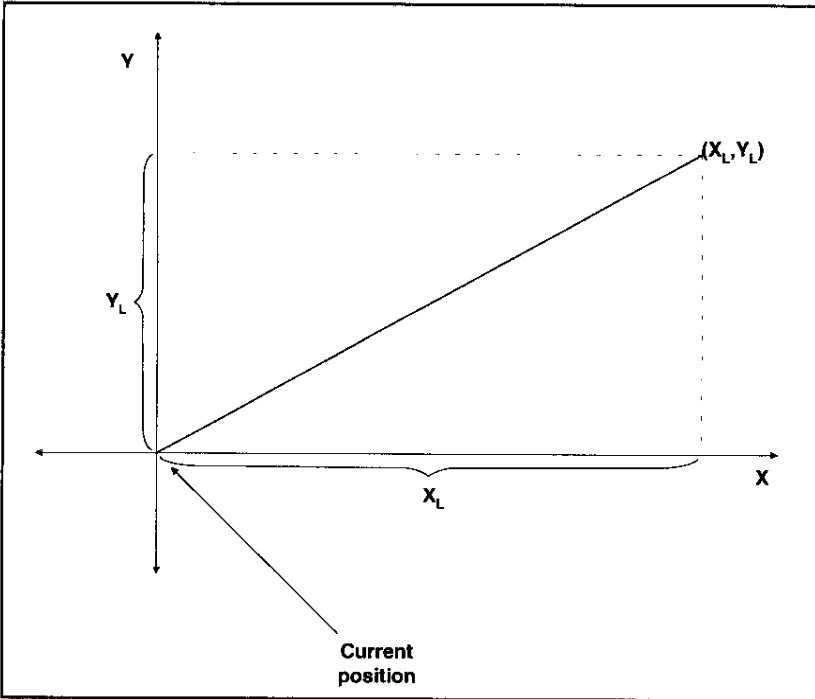
Using the LINE command

Example:

```
LINE (2,3), (300,200)    /* Draws a line to 300,200 */
```

LOOP (count#)

This command will repeat a command (or number of commands) count# times. The command(s) will be located between the LOOP and LOOPEND commands.

Format:

```
LOOP (count#)
      count# = number of repetitions
```

Example 1:

```
Loop (3)
IN 1
LOOPEND
```

Digital input port **1 is read 3 times** and the value **is displayed on the** monitor.

Example 2:

```
LOOP (3)  /* first loop -3  times */
LOOP (2)  /* second loop - 2 times */
HSPPMOVE +(1). 200
LOOPEND  /* end of second loop */
ECHO "Two moves completed"
LOOPEND  /* end of first loop */
```

Channel 1 moves 200 steps, 2 times, and then a message is displayed. This is repeated 3 times.

The above syntax is called 'nested looping'. PROG839.EXE allows 10 levels of nesting.

MANUAL

This command will turn PROGg39.EXE into manual mode. In this mode you can adjust the stepping motor position or the parameters for every channel manually. The left and right arrow keys are used to step between the channels. When you have finished adjusting the parameters/position, press [ESC]. The parameters will be saved and used as the default settings.

Format:

```
MANUAL
```

OUT *port #. value*

The PCL-839 has two digital output ports, 0 and 1. Digital values can be output to external devices on these ports. This command is used to write a value to one of the ports.

Format:

```
 OUT port#,value
   port# = port number (0 or 1)
   value = value to be output on port
```

Example:

```
OUT 0,OX33
OUT 1.55
```

Hex 33 (00110011) will be output to port 0, and then decimal 55 (00110111) will be output to port 1.

RUN

Sometimes it is necessary to output commands to different channels at the same time. The RUN command allows the delayed commands to be executed simultaneously when 'RUN' is interpreted.

Format:

```
   RUN
```

Example:

```
HSPPMOVE  +(1,2),4000
LSPMOVE - (3 ) . 2000
RUN
```

HSPPMOVE instructs channels 1 and 2 to move 4000 steps in the (+) direction. LSPMOVE instructs channel 3 to move 2000 steps in the (-) direction These commands are stored, and when the RUN command is interpreted, it sends these commands to the different motors simultaneously

SET (CH# [,CH# [, CH#]]), (FL, FH, AD)
 This command is used to set a channel's FL, FH and AD parameters.

Format:

```
SET (CH# [,CH# [,CH#]]), (FL, FH, AD)
```
    FL(FL speed):l to 16382 pps
    FH(FH speed): 1 to 16382 pps
    AD (acceleration/deceleration rate): 2 to 1023

Example:
```
    SET (1. 2, 3). (400.3000,300)
```

This sets all three channels' FL to 400, FH to 3000 and AD to 300.

SETMODE (CH# [,CH# [,CH#]]),mode
This command sets the mode for the PCL-839 - either pulse mode or
direction mode. Direction mode means that two signals are used to control
the stepping motor - one for direction and one for step. In pulse mode only
one signal will be applied to either (+) direction or (-) direction.

Format:

```
SETMODE (CH# [,CH# [,CH#]]),mode
```
mode = DIR (direction mode) or PUS (pulse mode)

Example:
```
SETMODE (1.2). DIR /* sets oh 1 & 2 for direction
                                        mode.*/
SETMODE (3). PUS /* sets  channel 3 for pulse mode.*/
```

SLOWDOWN (CH# [,CH# [,CH#]])

This command causes a channel (or more) to ramp down its output frequency to FL speed.

Format:

```
SLOWDOWN (CH# [,CH# [,CH#]])
```

Example :

```
HSPCMOVE +(1,2)    /* delayed command */
RUN
WAITTIME (1000)    /* delay 1 second */
SLOWDOWN (2)
```

The commands above set channels 1 & 2 to high continious speed, wait for 1 second and then ramps the output of channel 2 down to FL speed.

SLSTOP (CH# [,CH# [,CH#]])

This command tells a channel (or more than one) to slow down to FL speed and then stop.

Format:

```
SLSTOP (CH# [,CH# [,CH#]])
```

Example :

```
HSPCMOVE +(2)
RUN
WAITTIME (2000)
SLSTOP (2)
```

The above commands instruct channel 2 to rotate in the (+) direction, and after a 2 second delay tells channel 2 to slow down to FL, and then stop.

STOP (CH# [,CH# [,CH#]])

This commands stops a channel (or channels) immediately.

Format:

```
STOP (CH# [,CH# [,CH#]])
```

Example :

```
LSPCMOVE +(1,2)
HSCPMOVE +(3)
RUN
WAITTIME (1000)
STOP (1,2)
SLSTOP (3)
```

Channels 1 and 2 stop immediately, channel 3 slows down to FL, then stops.

STATUS

This command displays the current status of all channels on the screen.

WAITDI port#, value

This command reads digital input port *port#*, and compares the value with *value*. Processing halts until these values are equal.

Format:

```
WAITDI port#, value
```
   port# = 1 or 0,
   value is the desired value.

Example :

```
HSPCMOVE +(1)
RUN
WAITDI 0,0X33
SLSTOP (1)
```

Channel 1 is set to high speed continious mode until digital input port 0 is equal to 33 (Hex), then slows down and stops.

WAITKEY

No further commands will be interpreted until a key is pressed on the PC's keyboard.

Example :

```
LSPCMOVE -(2)
RUN
WAITKEY
STOP (2)
```

Channel 2 is set to low speed mode, and stops when a key is pressed on the keyboard.

WAITRDY (CH#[,CH#[,CH#]])

This command waits until the CH# is ready. It checks the PCL-839's status and waits until CH# bit 6 of STATUS 0 = 0.

Format:

```
WAITRDY (CH#[,CH#[,CH#]])
```

Example :

```
LSPORG +(1,2,3)
RUN
WAITRDY (1,2,3)
```

Channels 1, 2 and 3 is instructed to go to "ORIGIN" point, and WAITRDY waits until all the channels's "ORG" is on.

WAITTIME (ms)

This command delays any further commands by *ms* milliseconds.

Format:

```
WAITTIME (ms)
```
   ms = delay time in milliseconds.

Example :

```
WAITTIME (500)
STOP (1)
```

Delay for 0.5 seconds, then stop channel 1.

LSPCMOVE *dir*(CH#[,CH#[,CH#]])

This command sets channel CH# at FL speed in the *dir direction* until the STOP command is executed.

Format:

```
LSPCMOVE dir (CH#[,CH#[,CH#]])
     dir = direction
     CH# is channel number
```

Example :

```
LSPCMOVE +(3)
LSPCMOVE -(2)
RUN
WAITKEY
STOP (2,3)
```

HSPCMOVE *dir*(CH#[,CH#[,CH#]])

This command instructs channel CH# to ramp up from FL to FH, and continue stepping until the STOP, SLSTOP or SLOWDOWN command is executed. The movement is in *dir* direction.

Format:

```
HSPCMOVE dir(CH#[,CH#[,CH#]])
     dir = direction
     CH# = channel number
```

Example :

```
HSPCMOVE +(1)
HSPCMOVE -(2,3)
RUN
WAITTIME (1000)    /* continious move for one second */
SLSTOP (1,2,3)     /* deceleration stop */
```

LSPORG dir(CH#[,CH#[,CH#]])

Channel no CH# is set to low speed (FL) until limit switch ORG is
on.

Format:

```
LSPORG dir(CH#[,CH#[,CH#]])
```

Example :

```
LSPORG +(1,2)
LSPORG -(3)
RUN
WAITRDY(1,2,3)
```

HSPORG dir(CH#[,CH#[,CH#]])

This command instructs channel CH# to ramp up to FH speed,
from FL speed, and to continue moving at FH speed until limit
switch ORG is on. Channel CH# will decelerate to FL when SD
goes active and will stop when ORG goes active.

Format:

```
HSPORG dir(CH#[,CH#[,CH#]])
```

Example :

```
HSPORG +(2)
RUN
WAITRDY (2)
```

LSPPMOVE dir(CH#[,CH#[,CH#]]),step#

Channel CH# is instructed to move *step#* steps in the *dir* direction at FL speed.

Format:

```
LSPPMOVE dir(CH#[,CH#[,CH#]]),step#
    step# = number of steps
```

Example :

```
LSPPMOVE +(1,2),2000
LSPPMOVE -(3),4000
RUN
```

Channels 1 & 2 moves 2000 steps in the + direction at FL speed, and channel 3 moves 4000 steps in the - direction.

HSPPMOVE dir(CH#[,CH#[,CH#]]),step#

Channel CH# ramps up from FL to FH and moves in the *dir* direction, at FH speed. Channel CH# then decelerates and stops. The total number of steps taken is *step#*.

Format:

```
HSPPMOVE dir(CH#[,CH#[,CH#]]),step#
    step# = total steps
```

Example :

```
SET (1),(200,3000,400)    /* FL = 200, FH=3000, AD=400 */
HSPPMOVE +(1),5000
RUN
```

Channel 1 ramps up from 200 to 3000 pps. It continues stepping at 3000 pps and then ramps down to standstill. The total number of steps taken for this action is 5000 steps.

4

PCL-839 Software Library

Introduction

On the floppy disk that came with your PCL-839 card, there are 'C' library files. These libraries were developed in 'Turbo C', and you should be able do develop your own stepping motor applications (in 'C') using these files. The source code for the programming library ('LIB839.C') can also be found on the floppy disk. This enables you to recompile the libraries for any 'C' compiler (although some minor changes may be necessary).

The following sections describe the files and functions that will assist you when you write applications for the PCL-839.

The 'PCL839.H' Header File

To be able to use the functions contained in the software library, you have to include this header file in your source program (#include "PCL839.H"). This file contains the headers (Prototypes) for all the functions defined in LIB839CX.LIR.

PCL839.H contains the following:

```
int base = 0x300 ; /* base address . default = 0x300
int puls_dir[3] ;  /* used by PCL-839~s function */
int PO = O :       /* DIO port #O (8bit)
int Pi = 1 ;       /* DIO port #1 (8-bit)
int P01 = 2 ;      /* DIO port #O h #1 *
int CH1 = 1 :      /* Channel #1
int CHZ = 2 ;      /* Channel #2
int CH3 = 3 ;      /* Channel #3
int CH1Z = 4 ;     /* Channel #1 h #2
int CH13 = 5 ;     /* Channel #1 & #3
int CHZ) = 6 :     /* Channel #Z & #3
int CH123 = 7 ;    /* Channel #1.#2 and #3
int P_DIR = 0 :    /* Positive (+) direction
int N_DIR = l ;    /* Negative (-) direction */
int FL = 0 :       /* FL speed */
int FH = 1 ;       /* FH speed */
int DIR = 0 :      /* Direction mode */
int PUS = 1 :      /* Pulse mode */
int out_port(int portl_no . int value);
int in_port(int port_no);
```

```
int set_base(int b);
int set_mode(int ch , int mode);
int set_speed(int ch , int r1 , int r2 , int r4);
int status(int ch);
int org(int ch , int dir1 , int speed1 ,
         int dir2 , int speed2 ,
         int dir3 , int speed3 );
int cmove(int ch , int dir1 , int speed1 ,
          int dir2 , int speed2 ,
          int dir3 , int speed3 );
int pmove(int ch , int dir1 , int speed1 , long step1 ,
          int dir2 , int speed2 , long step2 ,
          int dir3 , int speed3 , long step3 );
int stop(int ch);
int sldn_stop(int ch);
int waitrdy(int ch);
int slowdown(int ch);
int line(int ch_plan, int x, int y);
int arc(int ch_plan, int dir, long X1, long Y1, long X2,
  long Y2);

/* NOTE :  port_no = 0 , 1 , 2
     ch      = 1 , 2 , 3 , 4 , 5 , 6 , 7  except
               function status_ch(ch) that ch = 1 , 2 , 3
  ch_plan = 4, 5, 6
  dir = 0 , 1
  speed = 0 , 1
  mode = 0 , 1
*/
```

'PCL839CX.LIB' Library file

Four library files have been included with the software. Although all these libraries contain the same functions, they have been compiled for different memory models:

LIB839CS.LIB   'Small model' library
LIB839CM.LIB 'Medium model' library
LIB839CC.LIB 'Compact model' library
LIB839CL.LIB 'Large model' library.

If you are using 'Turbo C' - one of the above libraries have to be included in the 'C' project-file that you are working on.

Function Call Descriptions

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

There are 15 functions in the PCL839 library. They are the following :

Function 1: set_base

This function sets the base address of the PCL-839. This enables the use of multiple PCL839s, if you require to do so.

Prototype:

```
int set_base(int BASE)
```

Parameters:

base   base address of PCL839 card

Return values:

0 : No error occurred when setting the base address

-1: An error occurred when setting the base address

Example(s):

```
int error_code = set_base(0x2C0);
```

Function 2: set_mode

This function sets the output mode for a channel, or group of channels.

Prototype:

```
int set_mode(int ch, int mode);
```

Parameters:

ch	channel number	1 for channel 1
		2 for channel 2
		3 for channel 3
		4 for channels 1 & 2
		5 for channels 1 & 3
		6 for channels 2 & 3
		7 for channels 1, 2 & 3
mode	0 for Direction (one-pulse) mode	
	1 for Pulse (two-pulse) mode	

Return values:

    0 : No error occurred

    -1: An error occurred

Example(s):

```
error_code = set_mode(CH12, DIR);
error_code = set_mode(CH3, PUS);
```

Channels 1 and 2 is set to direction mode, and channel 3 is set to pulse mode.

Function 3: set_speed

This function sets the low-speed pulse output frequency, high-speed pulse output frequency and acceleration/deceleration rate for a channel.

Prototype:

```
int set_speed(int ch,int FL,int FH,int AD);
```

Parameters:

ch	channel number	(See Function 2).
FL	Low-speed frequency	1 - 16382 pps.
FH	High-speed frequency	1 - 16382 pps.
AD	Acceleration/decel. rate	2 - 1023

Return values:

    0 : No error occurred

    -1: An error occurred

Example(s):

```
error_code = set_speed(CH123, 400, 3000, 500);
```

Channels 1, 2 and 3's FL is set to 400 pps, FH is set to 3000 pps and AD is set to 500 pps^2.

Function 4: status

This function reads and return the status of a channel.

Prototype:

```
int status(int ch);
```

Parameters:

ch   channel number   1 - channel 1
                                     2 - channel 2
                                     3 - channel 3

Return values:

-1: An error occurred

Other : The high byte will contain the value of 'Status 1' and the low byte the value of 'Status 0'.

Example(s):

```
int channel_status;
channel_status = status(CH1);
```

Function 5: stop

This function stops channel ch.

Prototype:

```
int stop(int ch);
```

Parameters:

ch   channel number   (See Function 2).

Return values:

0 : No error occurred
-1: An error occurred

Example(s):

```
int error_code;
error_code = stop(CH123);
```

Channels 1, 2 and 3 are stopped.

Function 6: slowdown

This function ramps the output frequency of channel(s) ch down to FL.

Prototype:

```
int slowdown(int ch);
```

Parameters:

ch   channel number   (See Function 2).

Return values:

  0 : No error occurred

  -1: An error occurred

Example(s):

```
int error_code = slowdown(CH23);
```

Channels 2 and 3's is ramped down to FL.

Function 7: sldn_stop

This function ramps the output frequency of channel(s) ch down to FL.

Prototype:

```
int sldn_stop(int ch);
```

Parameters:

ch   channel number   (See Function 2).

Return values:

  0 : No error occurred

  -1: An error occurred

Example(s):

```
error_code = sldn_stop(CH12);
```

Channels 1 and 2's is ramped down to FL, and then stopped.

Function 8: waitrdy

This function checks the 'Status0' of channel(s) ch and waits until bit 6 of 'Status0' is 0

Prototype:

```
int waitrdy(int ch);
```

Parameters:

ch   channel number   (See Function 2).

Return values:

0 : No error occurred

-1: An error occurred

Example(s):

```
error_code = waitrdy(CH12);
```

A delay is caused until channels 1 and 2's 'Status0' is 0.

Function 9: out_port

This function outputs a value *value* to port *port_no*.

Prototype:

```
int out_port(int port_no, int value);
```

Parameters:

port_no digital output port number 0 for port 0 (DO7-0)

                              1 for port 1 (DO15-8)

                              2 for ports 0 & 1

If P01 is used (output to both ports), the high-byte of the word will be output to port 1, and the low-byte of the word to port 0.

Return values:

0 : No error occurred

-1: An error occurred

Example(s):

```
error_code = out_port(P0, 0XAA);    /* 0XAA to port 0 */
error_code = out_port(P01,0X55AA); /* 0X55 to port 1 and
0XAA to port 0 */
```

Function 10: in_port

This function reads and returns the value of a digital input port.

Prototype:

```
int in_port(int port_no);
```

Parameters:

port_no D/I port number   0 for port 0 (DI7-0)
                                    1 for port 1 (DI15-8)
                                    2 for ports 0 & 1

If P01 is used (input from both ports), the high-byte of the returned value will contain the value of port 1, and the low-byte the value of port 0.

Return values:

-1: An error occurred

Other : Port Value

Example(s):

```
port_value  = in_port(P0);
port_value  = in_port(P01);
```

Function 11: org

This function returns all three channels to the 'ORIGIN' point. The direction and speed (frequency) that each channel has to use, are supplied.

Prototype:

```
int org(int ch,   int DIR1, int SPEED1,
        int DIR2, int SPEED2,
        int DIR3, int SPEED3);
```

Parameters:

ch	channel number	(See Function 2).
DIRn	channel n direction	0 = (+) and 1 = (-)
SPEEDn	channel n frequency	0 = FL and 1 = FH

Return values:

0 : No error occurred

-1: An error occurred

Example(s):

```
error_code = org(CH12, P_DIR, FL, N_DIR, FH, 0, 0);
```

Channels 1 and 2's is returned to origin point - channel 1 pulses at FL frequency in the (+) direction, and channel 2 pulses at FH frequency in the (-) direction. Channel 3 is ignored.

Function 12: cmove

This function starts channel(s) *ch* in continious mode. Channel 1 will move in DIR1 direction at SPEED1 speed, etc.

The channel(s) will stay in continious move mode until 'stop' or 'sldn_stop' is executed.

Prototype:

```
int cmove(int ch, int DIR1, int SPEED1,
        int DIR2, int SPEED2,
        int DIR3, int SPEED3);
```

Parameters:

ch	channel number	(See Function 2).
DIRn	channel n direction	0 = (+) and 1 = (-)
SPEEDn	channel n frequency	0 = FL and 1 = FH

Return values:

0 : No error occurred

-1: An error occurred

Example(s):

```
error_code = cmove(CH2,0,0,P_DIR, FH, 0, 0);
```

Channel 2 is placed in continious move mode, and pulses at FH in the (+) direction.

Function 13: pmove

This function starts channel(s) *ch* in continious mode, for a certain amount of steps. Channel 1 will move in DIR1 direction at SPEED1 speed and will stop when it has completed STEP1 steps, etc.

Prototype:

```
int pmove(int ch,int DIR1,int SPEED1,long STEP1,
          int DIR2, int SPEED2, long STEP2,
          int DIR3, int SPEED3, long STEP3);
```

Parameters:

ch   channel number (See Function 2).

DIRn   channel n direction   0 = (+) and 1 = (-)

SPEEDn channel n frequency   0 = FL and 1 = FH

STEPn channel n steps   max. long.

Return values:

0 : No error occurred

-1: An error occurred

Example(s):

```
error_code = pmove(CH123, P_DIR, FL, 2000, P_DIR, FH, 3000,
N_DIR, FH, 2000);
```

Channel 1 moves in (+) direction at speed FL for 2000 steps,
Channel 2 moves in (+) direction at speed FH for 3000 steps and
Channel 3 moves in (-) direction at speed FH for 2000 steps.

Then they stop.

Function 14: line

If you are using two stepper motors simultaneously (in a plotter-type configuration), you can move to position (X, Y) from the current position, (0, 0). Both axis are measured in *steps*.

Prototype:

```
int line(int ch_plan, int X, int Y);
```
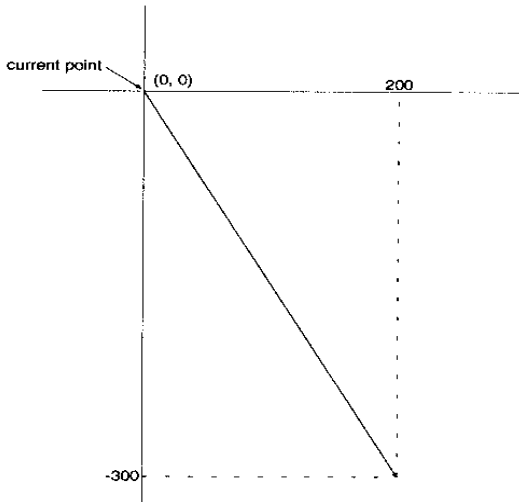
Parameters:

ch_plan channel numbers CH12 for channels 1&2
CH13 for channels 1&3
CH23 for channels 2&3

Return values:

0 : No error occurred
-1: An error occurred

Example(s):

```
error_code = line(CH23, 200, -300);
```

Function 15: arc

If you are using two stepper motors simultaneously (in a plotter-type configuration), you can "draw an arc" from (X1, Y1) to (X2, Y2)

Prototype:

```
int arc(int ch_plan, int dir, long X1,
                 long Y1, long X2, long Y2);
```

Parameters:

    ch_plan channel numbers CH12 for channels 1&2
                      CH13 for channels 1&3
                      CH23 for channels 2&3
    dir   direction0 for clockwise
                      1 for counterclockwise
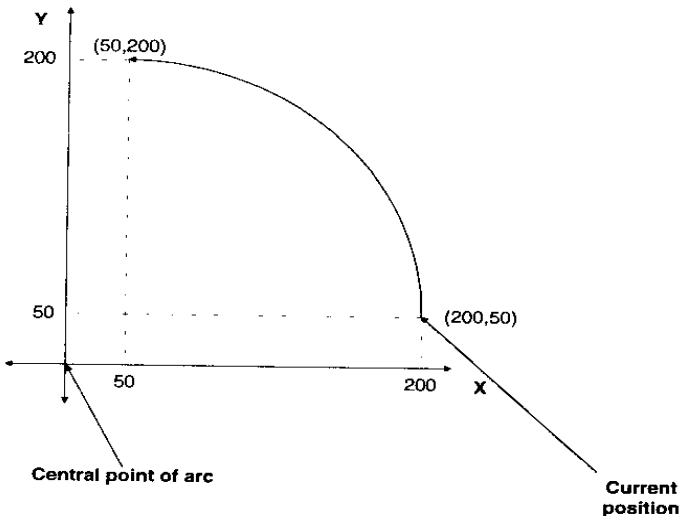   X1, Y1 coordinates of starting point
    X2, Y2 coordinates of final point

Return values:

    0 : No error occurred

    -1: An error occurred

Example(s):

```
error_code = arc(CH13, 1, 200, 50, 50, 200);
```

5
Register Programming

PCL-839 Registers

Several registers are used to control the PCL-X39. The PCL-839 uses these registers to store commands, speed, mode, number of pulses etc. The following sections describe these registers in detail.

RO : Down-Counter (18 bits)

The down-counter counts down when a pulse is output in manual mode, origin mode or preset mode. If the counter is stopped in operation mode, counting ceases. If a pulse is output when the counter has reached 0, the counter reverts to its maximum number (3FFFF in Hex, 262143 in decimal).

The counter value can be read at any stage - in operation or during standstill. When reading the value in operation, two quick reads must be done before the next pulse changes the value of the counter. Compare the two values - if they are the same then this is the true number of residual pulses.

In preset mode you set the required number of pulses on the counter. The counter counts down when a pulse is output and pulse generation will stop when the counter reaches O. The starting range is 00001 (hex) to 3FFFF (hex) (1 to 262143 in decimal notation). If the counter is set to O when operation is started, no pulse generation will occur. At that time the operation flag will indicate the halt condition, but the *INT signal is not output*.

If counting is interrupted by a deceleration-stop or reset command, the current counter value is stored, and counting will continue as soon as the start-command is received. As the counter will be at O when operation is complete, it is necessary to supply an initial value every time preset mode is started.

PCL-839 User' s Manual

R1 : FL Register (13 bits)

This register is used to set the FL (initial/low) speed. When started in high-speed mode, the generator starts at FL and ramps up to reach FH (Final/High speed). If the deceleration-stop command is received during high-speed operation, it ramps down to FL speed and then stops. *Make sure that you set a FL speed.*

The range for FL is 1 to 8191 (0001 to 1FFF in hexadecimal notation). Relation between a set value and the output pulse frequency vanes according to the value of R7 (multiplier register).

R2: FH Register (11 bits)

This register is used to set the FH speed. The range for FH is also 1 to 8191 (0001 to 1FFF in hexadecimal notation). The relation between a set value
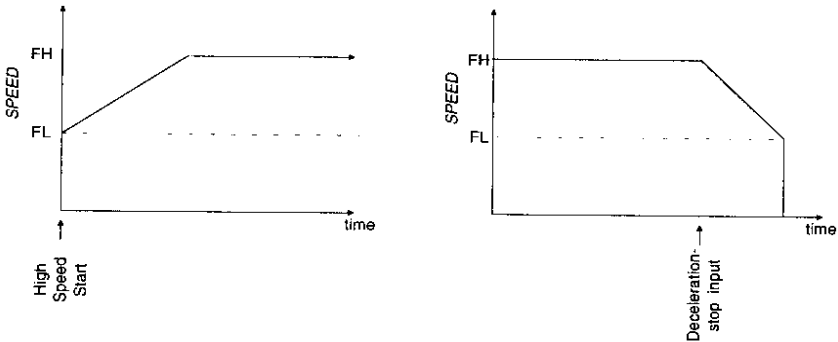
and the output pulse frequency also varies according to the value of R7 (multiplier register).

R4:Acceleration (ramping-up)/ Deceleration tramping-down) Rate Register (10 bits)

This register is used to set acceleration and deceleration characteristics. During high-speed mode, the generator starts at FL and accelerates to FH.

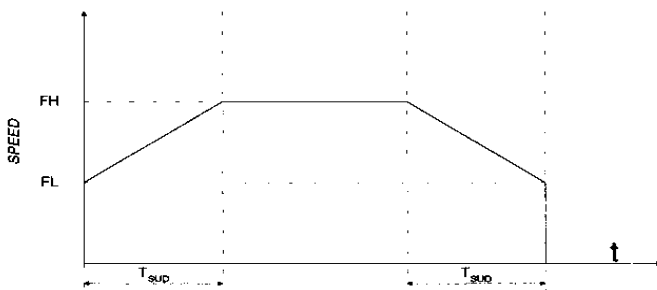If the reference clock frequency is (TCLK)[sec], T_{SUD}(the time required for the ramping-up/ramping-down) is

$$T_{SUD} = [(R2)-(R1)] \times (R4) \times (T_{CLK})[sec].$$

Alternatively, if the ramping-up/ramping-down time is known, R4 can be calculated as

$$R4 = T_{SUD}/([(R2)-(R1)] \times (T_{CLK}))$$

The range forR4 is 002 (hex) to 3FF (hex) (2 to 1023 in decimal).

Note : for the PCL-839, $T_{CLK} \cong 203ns$

R6 :Ramping-down Point Register (16 bits)

During high-speed operation, the value of the down-counter is compared with the value of this register. As soon as the value of the counter is less than the value of this register, ramping-down will start. If the value of R6 is higher than the down-counter when high-speed mode starts, ramping-up will not occur and the pulse generation will proceed at FL.

The range forR6 is 0001 (hex) to FFFF (hex) (1 to 65535 in decimal).
The ramping-down point is set in pulses.

* Setting of the ramping-down point

If "automatic setting of ramping-point" is selected for the output mode, no setting is required for R6. If automatic setting is not selected, the value for R6 has to be calculated and written in the register.

When determining the ramping-down point, the FL frequency, the FH frequency and the deceleration rate has to be taken into account. If an improper value is set, pulse output may be terminated halfway during ramping-down (Fig. A) or may continue after ramping-down, causing longer FL speed operation (Fig. C).
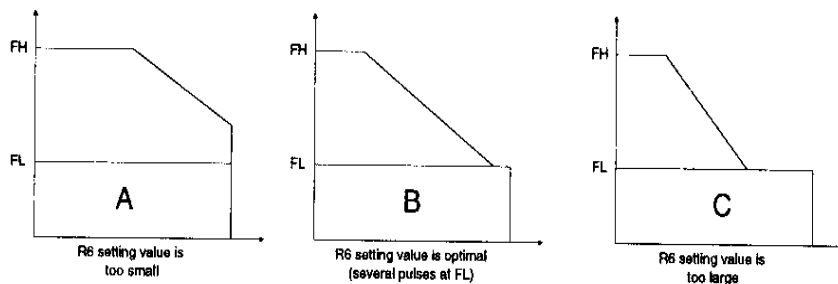
A ramping-down point is set based on the number of pulses output **during**
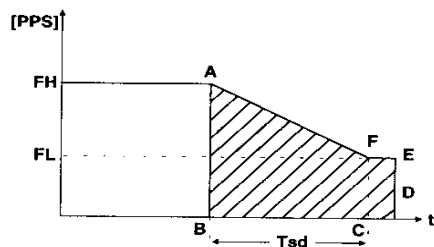
ramping-down. Therefore the area marked by oblique lines in the chart below is the number of pulses to be calculated. FL and FH are the output pulse frequencies.

Tsd [sec], the time required for the deceleration is
$T_{sd} = [(R2)-(Rl)] \times (R4)/(CLOCK)$ (1)
    where CLOCK = 4.9152 MHz
The relationship between the set value on speed register (Rf) and output frequency (F [PPS]) is
$F = (Rf) \times (CLOCK)/[8912 \times (R7)]$ (2)

Therefore, FL output frequency FL [PPS] and FH output frequency FH [PPS] are

$$FL = (Rl)x(CLOCK)/[8192x(R7)1 \quad (3)$$

$$FH = (R2)x(CLOCK)/[8192x(R7)1 \quad (4)$$

P_{sd}, the number of pulses during T, [sec] is represented by the area of the trapezoid A-B-C-F

$$P_{sd} = \{[(FL) + (FH)] \times T_{sd} \}/2 \qquad (5)$$

Substitute equations (1), (3) and (4) into equation (5)

$$P_{sd} = \{[(R2)^2 - (R1)^2]x (R4)\} / \{16384 \times (R7)\}$$

When outputting 5 pulses at FL speed after the completion of the ramping-down, the set value of the ramping-down register point register (R6) is

$$(R6) = P_{sd}+5$$

$$(R6) = \{[(R2)^2 - (R1)^2]x (R4)\} / \{16384 \times (R7)\} + 5$$

R7:Multiplier Register (10 bits)

For the speed registers, Fl and R2, a number of steps can be selected (1 to 8191). This register (R7) is used to assign an output frequency for one step. The reference clock inputted through the CLOCK terminal is divided and multiplied by the variable frequency divider and the frequency multiplier, and then outputted to the PULSE OUTPUT terminal. When a set value on the speed register is Rf (where Rf is a value set at R1 and R2), the frequency outputted at the PULSE OUTPUT terminal is

$$Fpout = \{(\text{Reference clock freq. [Hz]} \times (Rf)) / (8912 \times (R7)\}$$
$$= (Rf) \times \{(\text{Reference clock freq.})\} / \{8912 \times (R7)\}$$

When (reference clock)/ [8192 x (R7)1= 1 ... Ix mode

When (reference clock)/ [8192 x (R7)1= 2 ... 2x mode

For the PCL-839, the reference clock frequency is 4.9152 [MHz], Therefore

(R7)= 600   (=258 hex)   1x mode

(R7)= 300   (=12C hex)   2x mode

The setting range is 002 (hex) to 3FF (hex), which corresponds to 2 to 1023 in decimal notation. The smaller the set value, the higher the output frequency.

Programming the PCL-839

The PCL-839 stores a selected command in a buffer. This command remains there until a new command is received. The only command that can be RESET, is the 'starting mode' command.

I/O Register Control format.
The following table depicts the PCL-839's register I/O address map.

	WRITE	READ
BASE + 0	Channel 1 command buffer	Channel 1 status
BASE + 1	Channel 1 data buffer 0-7	Channel 1 data 0-7 of R0, R6
BASE + 2	Channel 1 data buffer 8-15	Channel 1 data 8-15 of R0, R6
BASE + 3	Channel 1 data buffer 16-17	Channel 1 data 16,17 of R0, R6 & STATUS
BASE + 4	Channel 2 command buffer	Channel 2 status
BASE + 5	Channel 2 data buffer 0-7	Channel 2 data 0-7 of R0, R6
BASE + 6	Channel 2 data buffer 8-15	Channel 2 data 8-15 of R0, R6
BASE + 7	Channel 2 data buffer 16-17	Channel 2 data 16,17 of R0, R6 & STATUS
BASE + 8	Channel 3 command buffer	Channel 3 status
BASE + 9	Channel 3 data buffer 0-7	Channel 3 data 0-7 of R0, R6
BASE + 10	Channel 3 data buffer 8-15	Channel 3 data 8-15 of R0, R6
BASE + 11	Channel 3 data buffer 16-17	Channel 3 data 16,17 of R0, R6 & STATUS
BASE + 12	DIGITAL OUTPUT 0-7 (PORT 0)	DIGITAL INPUT 0-7 (PORT 0)
BASE + 13	DIGITAL OUTPUT 8-15 (PORT 1)	DIGITAL INPUT 8-15 (PORT 1)
BASE + 14	IRQ control	IRQ status
BASE + 15	Same as BASE+14	Same as BASE+14

Command buffers : WR0, WR4 and WR8.

Each of the three channels have a command buffer which enables individual programming. Channel 1's command buffer is BASE + 0, Channel 2's is BASE + 4 and Channel 3's is BASE + 8. A command can be written to any of the three buffers, and the appropriate channel will respond to the command.

Register format
The register format is as follows :

Register format							
D7	D6	D5	D4	D3	D2	D1	D0
C1	C0						
| Mode |			Command				|

Selection modes
The two high-order bits of the command buffer specifies the command that will be executed. The remaining six bits contain command parameters. The command modes available are as follows :

Selection modes		
C1	C0	
0	0	Start-Stop Command selection
0	1	Operation Mode Select command
1	0	Register Select command
1	1	Output Mode select command

Command Parameters
The following sections describe all the available commands and their parameters in detail.

Commands

Start-Stop Command

```
7  6  5  4  3  2  1  0
```

| O | O | | | | | O | |

- Frequency Selection
  - 0 : Use FL frequency
  - 1 : Use FH frequency
- 0 : Constant speed mode
- 1 : High Speed mode
- 1 : Start/stop control
- 0 : Stop interrupt reset
- 1 : Stop interrupt enable

| O | O | * | 1 | O | O | O | O |
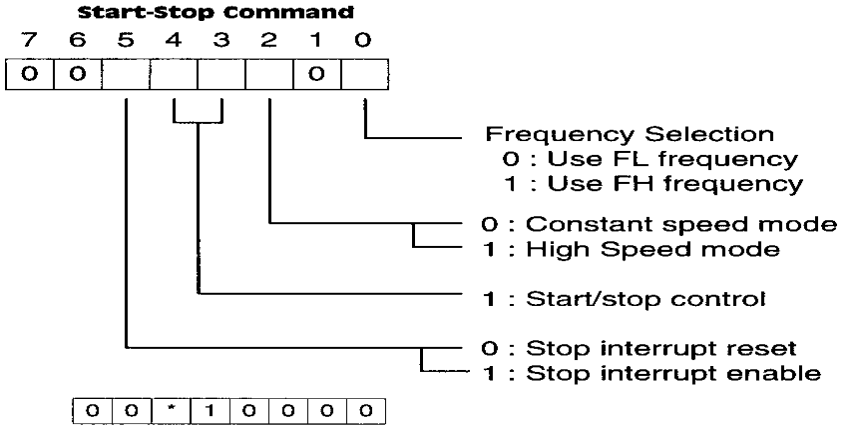
Constant speed operation with the FL register. Operates at the speed set for the FL register.

| O | O | * | 1 | 0 | O | O | 1 |

Constant speed operation with the FH register. Operates at the speed set for the FH register.

| O | O | * | 1 | O | 1 | O | 1 |

High speed operation with the FH register. Frequency ramps up halfway from the rate of the FL to that of the FH. During high-speed start this command lets the frequency ramp up/down to the rate of the FH speed.

| O | O | * | 1 | O | 1 | O | O |

Dual rate operation (ramping down). Frequency ramps down to the level of the FL.

* 0 (no output of INT signal at stop)
1 (output of INT signal at stop)

| 0 | 0 | * | 1 | 1 | 1 | 0 | 0 |

Decelerating stop (reset command is required after stop). Frequency ramps down to the rate of the FL, then stops.

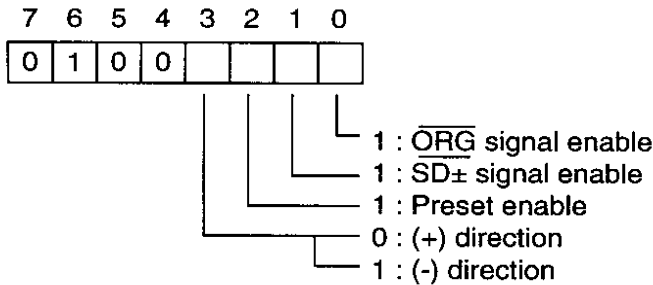| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Reset command. This stops pulse generation under any condition. If you start with the start-command, be sure to reset with the reset command before the next start. This gives $\overline{\text{INT}}$ signal and the start command has to be reset. Contents in registers R0 through R7 are not changed.

* 0 (no output of INT signal at stop)

　1 (output of INT signal at stop)

Operation Mode Select Command

```
  7   6   5   4   3   2   1   0
┌───┬───┬───┬───┬───┬───┬───┬───┐
│ 0 │ 1 │ 0 │ 0 │   │   │   │   │
└───┴───┴───┴───┴───┴───┴───┴───┘
                            └── 1 : ORG signal enable
                        └────── 1 : SD± signal enable
                    └────────── 1 : Preset enable
                └────────────── 0 : (+) direction
                └────────────── 1 : (-) direction
```

```
┌───┬───┬───┬───┬───┬───┬───┬───┐
│ 0 │ 1 │ 0 │ 0 │ * │ 0 │ 0 │ 0 │
└───┴───┴───┴───┴───┴───┴───┴───┘
```

Manual mode. Operation initiated in the start mode continues until the stop command is transferred.

```
┌───┬───┬───┬───┬───┬───┬───┬───┐
│ 0 │ 1 │ 0 │ 0 │ * │ 0 │ 0 │ 1 │
└───┴───┴───┴───┴───┴───┴───┴───┘
```

Origin return mode. Operation initiated in the start mode continues until the mechanical origin signal or stop command comes.

```
┌───┬───┬───┬───┬───┬───┬───┬───┐
│ 0 │ 1 │ 0 │ 0 │ * │ 1 │ 0 │ 0 │
└───┴───┴───┴───┴───┴───┴───┴───┘
```
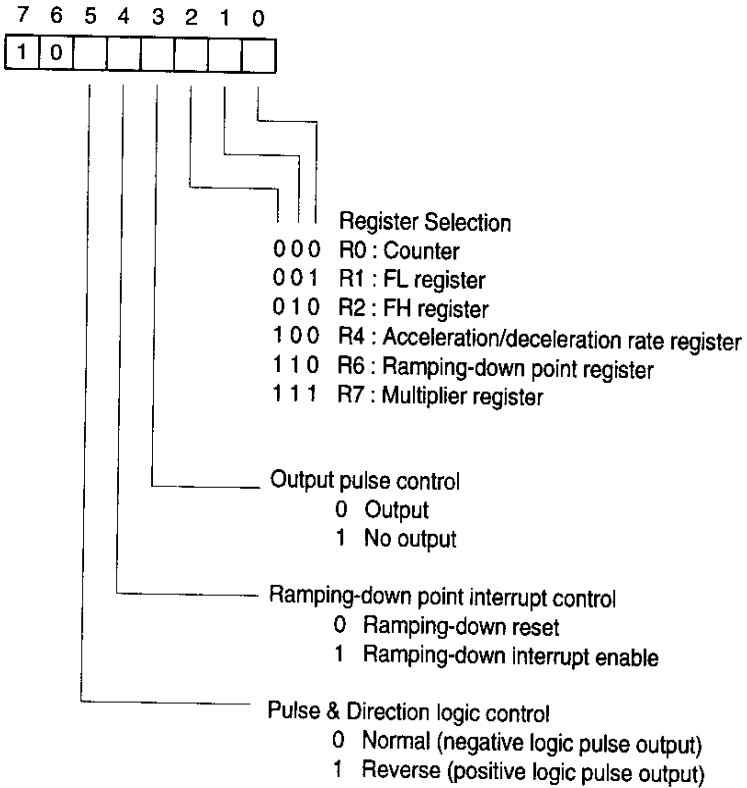
Preset mode. Operation initiated in the start mode, stops when the quantity set for register R0 is reached.

Operation in the high speed start mode, ramps down when the remaining quantity of the counter is less than the quantity set for register R6.
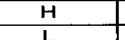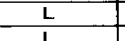
*: 0 (+) direction
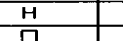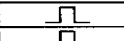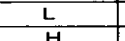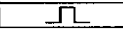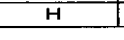
    1 (-) direction

Register Select Command

```
7  6  5  4  3  2  1  0
```
| 1 | 0 | | | | | | |

Register Selection
0 0 0   R0 : Counter
0 0 1   R1 : FL register
0 1 0   R2 : FH register
1 0 0   R4 : Acceleration/deceleration rate register
1 1 0   R6 : Ramping-down point register
1 1 1   R7 : Multiplier register

Output pulse control
　0　Output
　1　No output

Ramping-down point interrupt control
　0　Ramping-down reset
　1　Ramping-down interrupt enable

Pulse & Direction logic control
　0　Normal (negative logic pulse output)
　1　Reverse (positive logic pulse output)

- Register select code

    Selects an access register with 3 bits. After an undefined code (011 or 101) is selected, an attemp to write data in address 1 to 3 is unsuccessful.

- Output pulse control

    Enables operation without outputting a pulse from the PULSE/+dir (DIR/-dir) terminal pin. Since it does not influence *INT* and other signals, it can be used as a similation (machine lock) or a timer.

- Ramping-down point interrupt control

- PULSE/DIRECTION logic control

  PULSE/+dir and DIR/-dir output logic can be changed as follows

Setting	Direction	Directory Mode		Pulse Mode	
		PULSE/+dir	DIR/-dir	PULSE/+dir	DIR/-dir
0	(+)	⎍	H	⎍	H
	(−)	⎍	L	⎍	⎍
1	(+)	⎓	L	⎓	L
	(−)	⎓	H	L	⎓

:

● Kinds of registers and data bits

	Description	Bits	Read/Write
R0	Counter	18	R/W
R1	FL register	13	W
R2	FH register	13	W
R4	Acceleration/deceleration rate register	10	W
R6	Ramping-down point register	16	R/W (*)
R7	Multiplier register	16	W

Note : * Reading is possible when R6 is selected with a
register select command

Output Mode Select Command

```
7  6  5  4  3  2  1  0
┌──┬──┬──┬──┬──┬──┬──┬──┐
│ 1│ 1│  │  │  │  │  │ 0│
└──┴──┴──┴──┴──┴──┴──┴──┘
```
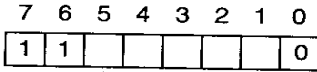
Pulse Output Mode
  0 : Directory mode
  1 : Pulse mode
Counter operation mode
  0 : Count by output pulse
  1 : Count stop
0 : Manual setting of ramping-down point
1 : Automatic setting of ramping-down point

1 : Ramping up/down stops.
$\overline{EL\pm}$, $\overline{ORG}$ signal input mode
  0 : Stops immediately
  1 : Ignores pulse inputs less than 4
      cycles in the reference clock.

Channel status buffers (RD0, RD4 and RD8)

There are a status buffer for each channel (status 0). These buffers are found at BASE +0, BASE +4 and BASE +8 for channel 1, channel 2 and channel 3 respectively. These buffers enable you to read the internal status of each channel, and also get certain information on input signals or conditions.
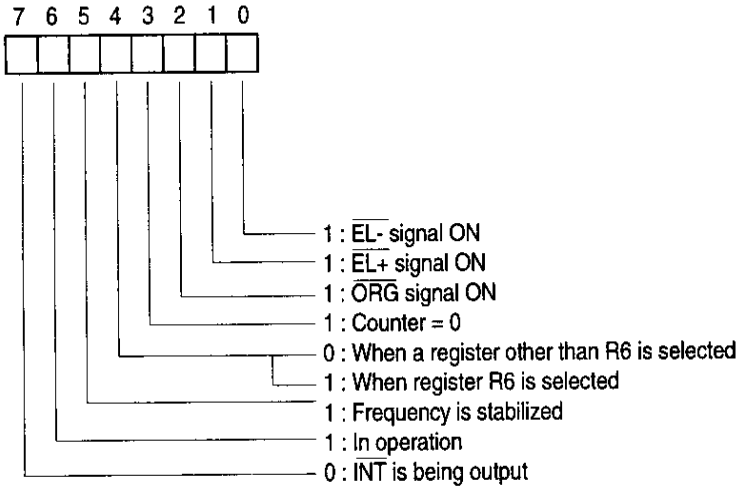
Status 0 bit configuration

Data buffers : WR1, WR5 and WR9

One data-buffer for each channel is found at BASE 1, BASE 5 and BASE 9, for channel 1, channel 2 and channel 3 respectively. When writing (output), these buffers contain data bits 0-7 of the respective channels.

Data buffers : RD1, RD5 and RD9

When data buffers WR1, WR5 and WR9 (see above) are read (input), these buffers contain data bits 0-7 of the respective channels, if registers R0 or R6 was selected (see register select command).

Data buffers : WR2, WR6 and WR10

One data-buffer for each channel is found at BASE+2, BASE+6 and BASE+10, for channel 1, channel 2 and channel 3 respectively. When writing (output), these buffers contain data bits 8-15 of the respective channels.

Data buffers : RD2, RD6 and RD10

When data buffers WR2, WR6 and WR10 (see above) are read (input), these buffers contain data bits 8-15 of the respective channels, if registers R0 or R6 was selected (see register select command).

Data buffers : WR3, WR7 and WR11

One data-buffer for each channel is found at BASE+3, BASE+7 and BASE+11, for channel 1, channel 2 and channel 3 respectively. When writing (output), these buffers contain data bits 16 &17 of the respective channels.

Data buffers : RD3, RD7 and RD11

When data buffers WR3, WR7 and WR11 (see above) are read (input), these buffers contain data bits 16 & 17 of the respective channels, if registers R0 or R6 was selected (see register select command). The bit definition is as follows :

Highest-order 2 bits of R0 (a register other than R6 was selected)

0 : Stop INT signal is being output (register R6 was selected)

0 : Down INT signal is being output (register R6 was selected)

1 : SD- signal ON

1 : SD+ signal ON

1 : Ramping-up in progress

1 : Ramping-down in progress

1 : R0 < R6

0 : PULSE signal output = H

1 : PULSE signal output = L
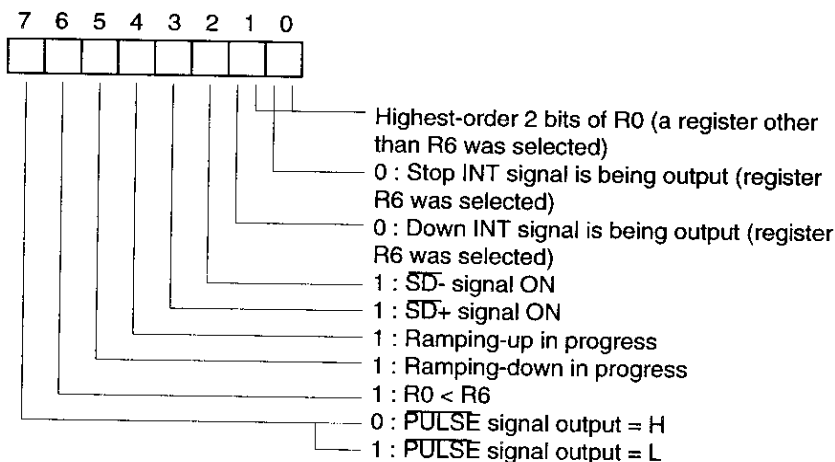
Status 1 bit configuration

Digital Outputs: WR12 and WR13

WR12 is the low byte of the digital output, and WR13 the high byte.
To write to these buffers, write to BASE12 and BASE13 respectively.
The bit definition is as follows :

bit	7	6	5	4	3	2	1	0
	DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

WR13

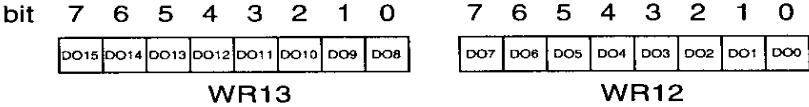| 7	6	5	4	3	2	1	0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

WR12

Digital inputs : RD12 and RD13

RD12 is the low byte of the digital intput, and RD13 the high byte. To read these buffers, address BASE+12 and BASE+13 respectively. The bit definition is as follows :

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

RD13

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

RD12
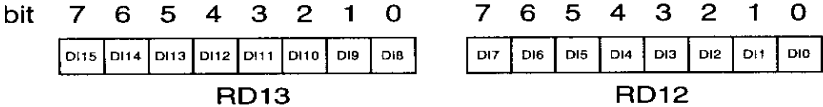
Interrupt control : WR14

WR14 is the interrupt control register for the PCL-839, and is found at BASE+14. Only one bit, b0, of the byte is used. When b0=1, interrupts are enabled, and when b0=0, disabled. If b0=1, and the channel-interrupt of the specific channel is also enabled (see Start-Stop command, bit 5), an interrupt will be generated when that channel reaches its 'stop position'.

bit	7	6	5	4	3	2	1	0
	---	---	---	---	---	---	---	IRQ EN

WR14

IRQEN
(BASE+14 BIT0)
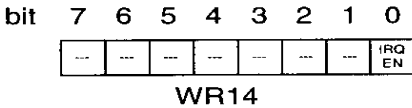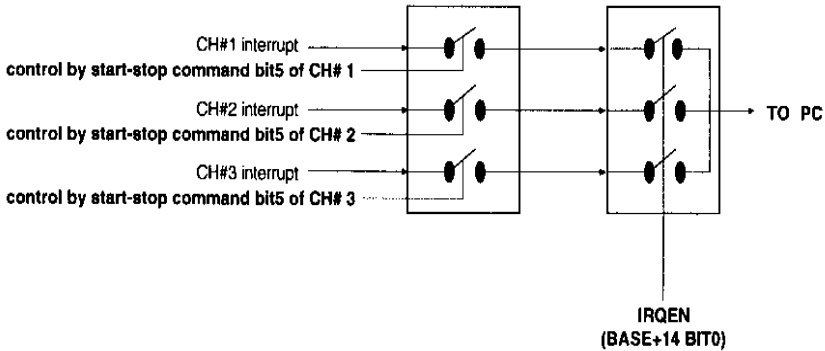
Interrupt path of the PCL-839

Interrupt Status register : RD14

RD14 is used to obtain the interrupt status for each channel. It is found at BASE+14, and contains the interrupt status for all three channels. When an interrupt occurs, this register can be read to determine which channel caused the interrupt. If the bit = 1, then an interrupt has occured for that channel. The bit configuration is as follows :

bit 7   6   5   4   3   2   1   0

| --- | --- | --- | --- | --- | IRQ CH#3 | IRQ CH#2 | IRQ CH#1 |
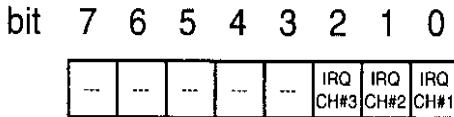
Note : When this register is read, bit 0 - bit 2 will be cleared.

Typical Operational Procedures

A. Initialization

B. Setting Speed Data

Select the FL Register → Command buffer <- 10000001 (81H)

Set FL data → Register bits 7-0 <- FL data, low-order 8 bits
Register bits 15-8 <- FL data, high-order 5 bits

Select the FH register → Command buffer <- 10000010 (82H)

Set FH data → Register bits 7-0 <- FH data, low-order 8-bits
Register bits 15-8 <- FH data, high-order 5 bits

Select the acceleration rate register → Command buffer <- 10000100 (84H)

Set acceleration rate data → Register bits 7-0 <- Acceleration rate data, low-order 8-bits
Register bits 15-8 <- Acceleration rate data, high-order 2 bits

Select the ramping-down point register → Command buffer <- 10000110 (86H)

Set ramping-down point data → Register bits 7-0 <- Ramping-down point data, low-order 8-bits
Register bits 15-8 <- Ramping-down point data, high-order 8 bits
(Setting is not required if automatic ramping-down setting mode is selected)

Select the multiplier register → Command buffer <- 10000111 (87H)

Set multiplier data → Register bits 7-0 <- Multiplier data, low-order 8 bits
Register bits 15-8 <- Multiplier data, high-order 2 bits

END

C. Constant Speed Preset Mode

f

Stop at pre-set count

t

| Execute the reset command | Command buffer <- 00001000 (08H) |

| Effect the preset mode and select the direction | For (+) direction : Command buffer <- 01000100 (44H)
For (-) direction : Command buffer <- 01001100 (4CH) |

| Select the counter | Command buffer <- 10000000 (80H) |

| Set the number of ouput pulses | Register bits 7-0 <- Data, low-order 8 bits
Register bits 15-8 <- Data, mid-order 8 bits
Register bits 23-16 <- Data, high-order 2 bits |

| Start | For FL : Command buffer <- 00010000 (10H)
For FH : Command buffer <- 00010001 (11H) |

The status (0) is read

Waiting for completion of operation

NO

Bit 6 = 0 ?

YES

Completion of operation

D. High Speed Preset Mode

f↑

Ramp down at set point

Stop at pre-set count

t

Execute the reset command	Command buffer <- 00001000 (08H)
Effect the ramping mode and select the direction	For (+) direction : Command buffer <- 01000100 (44H) For (-) direction : Command buffer <- 01001100 (4CH)
Select the counter	Command buffer <- 10000000 (80H)
Set the number of ouput pulses	Register bits 7-0 <- Data, low-order 8 bits Register bits 15-8 <- Data, mid-order 8 bits Register bits 23-16 <- Data, high-order 2 bits
Start	Command buffer <- 00010101 (15H)

The status (0) is read

Waiting for completion of operation

NO

Bit 6 = 0 ?

YES

Completion of operation

E. Constant Speed Continious Mode

Execute the reset command	Command buffer <- 00001000 (08H)

Effect the manual mode and set the direction	For (+) direction : Command buffer <- 01000000 (40H) For (-) direction : Command buffer <- 01001000 (48H)

Enter the start command	For FL : Command buffer <- 00010000 (10H) For FH : Command buffer <- 00010001 (11H)

Enter the immediate stop command	Command buffer <- 00001000 (08H) (reset command)

Completion of operation

F. High Speed Continious Mode

Execute the reset command — Command buffer <- 00001000 (08H)   (reset command)

Effect the manual mode and select the direction — For (+) direction : Command buffer <- 01000000 (40H)
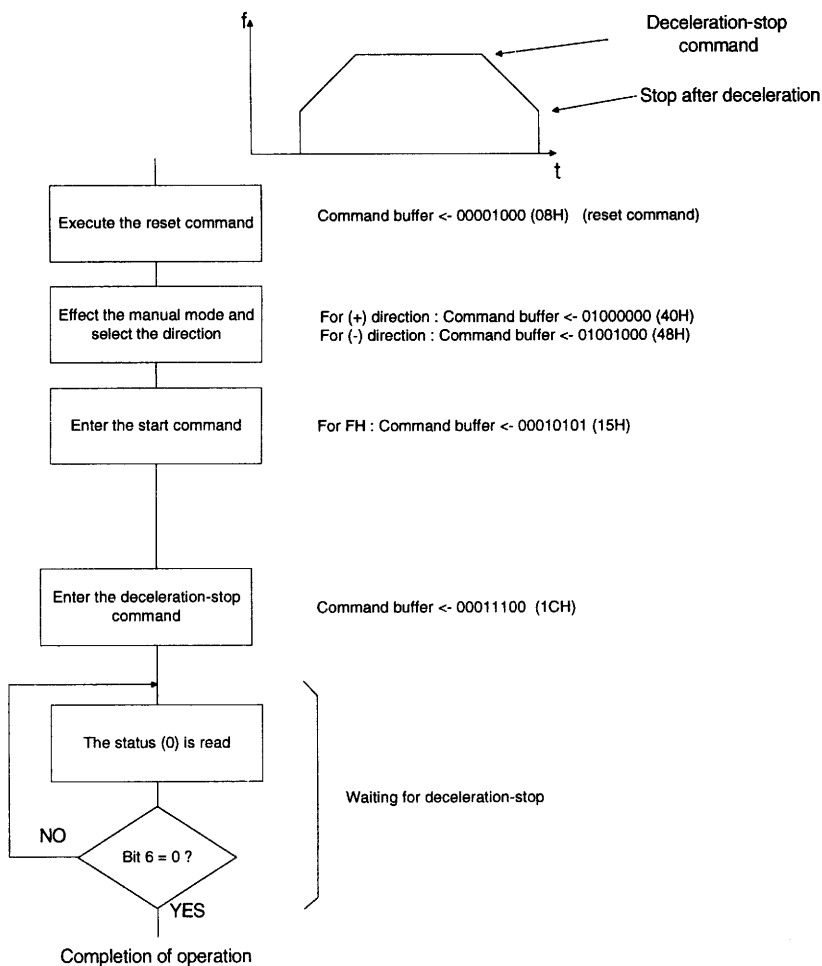For (-) direction : Command buffer <- 01001000 (48H)

Enter the start command — For FH : Command buffer <- 00010101 (15H)

Enter the deceleration-stop command — Command buffer <- 00011100 (1CH)

The status (0) is read

Waiting for deceleration-stop

NO — Bit 6 = 0 ?

YES

Completion of operation

G. Constant Speed Origin Return Mode

f

Stop by origin
switch "ON"

t

Execute the reset command

Command buffer <- 0001000 (08H) (reset command)

Effect the origin return mode and select the direction

For (+) direction : Command buffer <- 01000001 (41H)
For (-) direction : Command buffer <- 01001001 (49H)

Enter the start command

For FH : Command buffer <- 00010000 (10H)
For FL : Command buffer <- 00010001 (11H)

When the ORG signal is "ON", pulse output stops and bit 6 of status 0 becomes 0

The status (0) is read

Bit 6 = 0 ?

YES

Completion of operation

H. High Speed Origin Return Mode

Ramp down by SD switch "ON"

Stop by origin switch "ON"

f

t

| Execute the reset command | Command buffer <- 00001000 (08H) (reset command) |

| Effect the origin return mode and set the direction | For (+) direction : Command buffer <- 01000011 (43H)
For (-) direction : Command buffer <- 01001011 (4bH) |

| Enter the start command | For FH : Command buffer <- 00010101 (15H) |

When the SD limit switch of the same direction as the preset direction in "ON", the frequency ramps down to the rate of FL.

When the ORG limit switch is "ON", pulse output stops and bit 6 of status 0 becomes 0

The status (0) is read

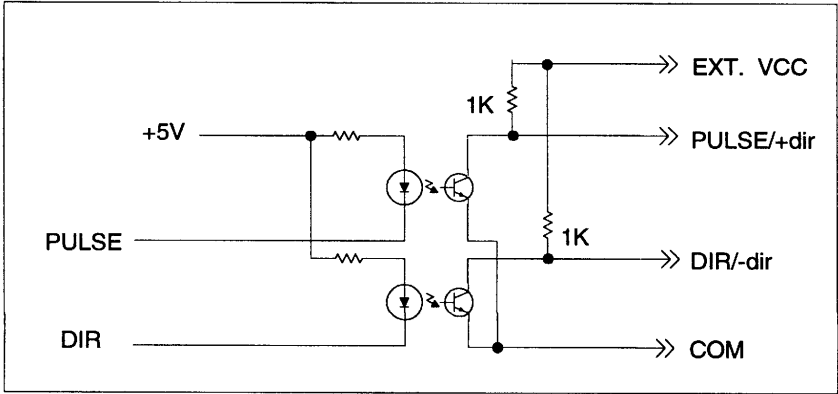Bit 6 = 0 ?

YES

Completion of operation

A
Diagrams

PCL-839 Jumper and Switch Layout
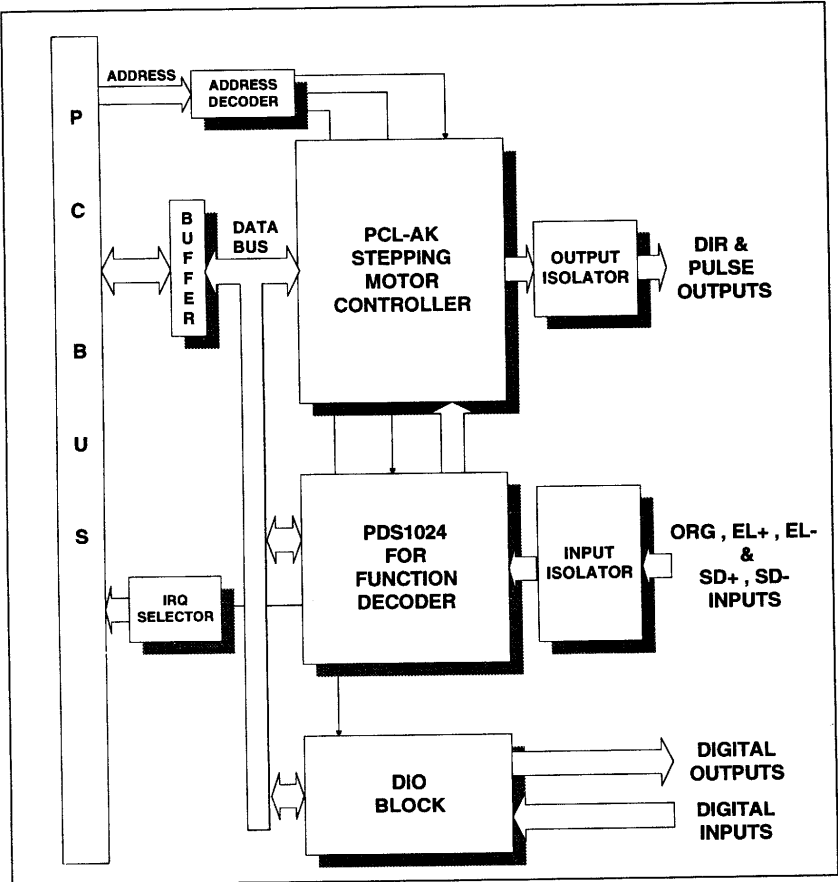
PCL-839 High Speed Motor Control Card
Block Diagram

Output Circuit Diagram

PCL-839 User's Manual
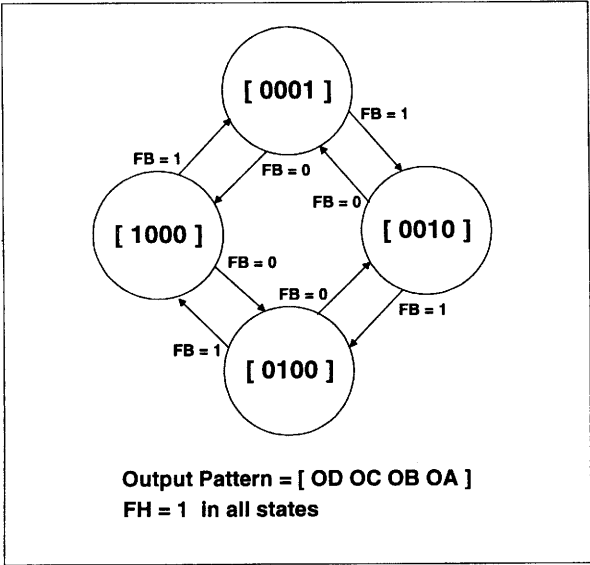
B
Simple Stepping Motor Driver
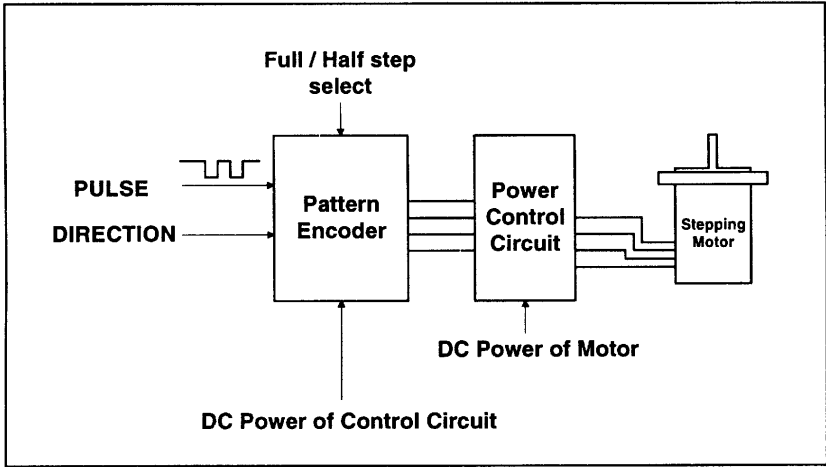
Simple Stepping Motor Driver Block Diagram
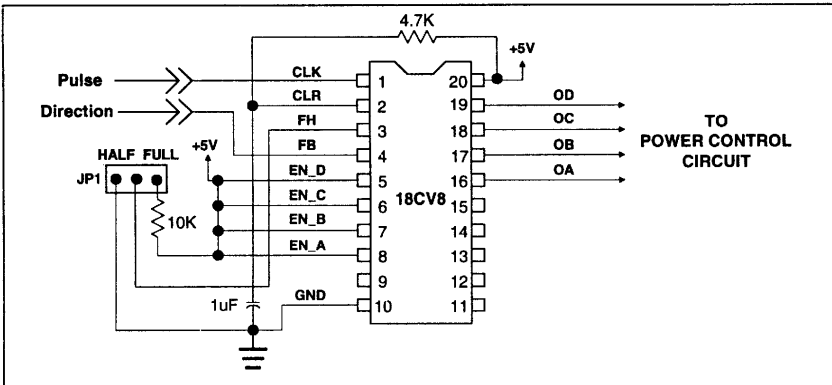
Pattern Encoder Connection

Output Pattern = [OD OC OB OA]
FH = 1 in all states

JP1 at Full Step Control

Output Pattern = [OD OC OB OA]
FH = 0 In all states

JP1 at Half Step Control

PCL-839 User's Manual

4-phase Stepping Motor Power Control Circuit
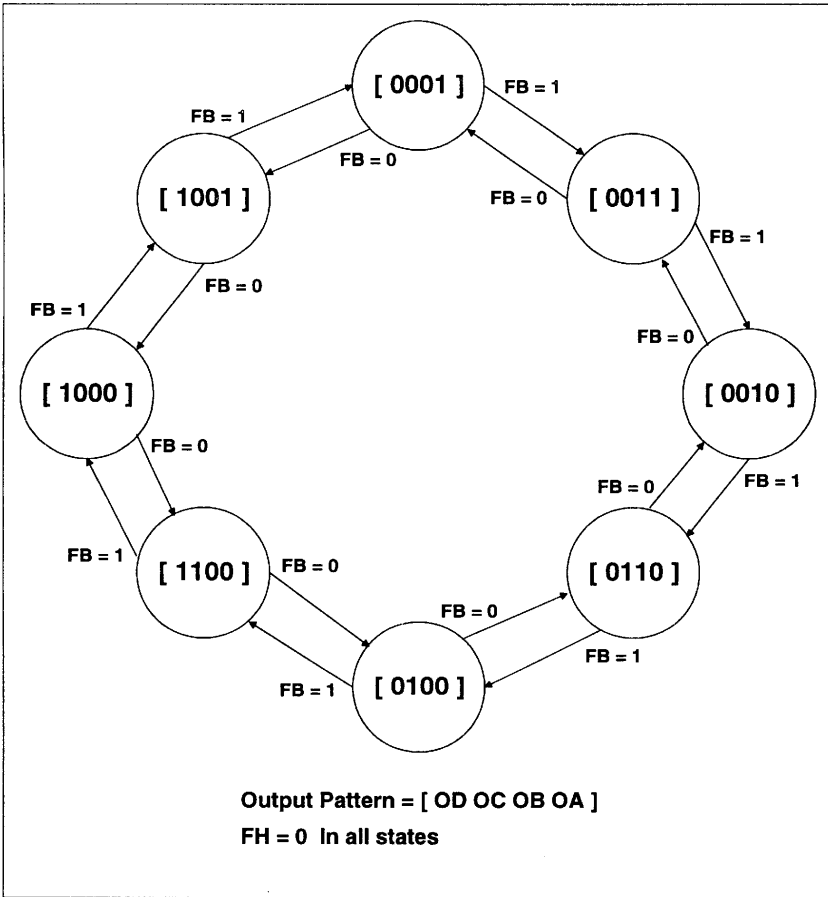
C
Utility Diskette Contents

PCL-839 Utility Diskette
Disk contents
The PCl-839 Utility Diskette contains the following files :

Directory \C

        MANUAL.C
        MANUAL.EXE
        I_MANUAL.EXE
        DEMO 1.C
        DEMO 2.C
        PCL839CS.LIB
        PCL839CM.LIB
        PCL839CC.LIB
        PCL839CL.LIB
        PCL839.H

Directory \INTERPRE

        PROG839.EXE
        CURVE.TXT
        SYNTEX.DOC

Program descriptions
PROG839.EXE
PROG839.EXE is a command interpreter for the PCL-839. You can
use this program to learn the PCL-839 function commands.

MANUAL.EXE
MANUAL.EXE is an utility program that enables you to manually control the
motor position, parameters and calibration of the motor position.

I_MANUAL.EXE

I_MANUAL.EXE's function is identical to MANUAL.EXE, except that I_MANUAL.EXE uses interrupts. When the PCL-839 finishes a command, the interrupt service routine will sound a bell.

PCL838CX.MB

When using 'C' to program the PCL-839, you can use various memory models. We have included libraries for the 'small', 'compact', 'medium' and 'large' memory models. The files are as follows:

    PCL839CS.LIB -'small' model library

    PCL839CS.LIB -'compact' model library

    PCL839CS.LTB -' medium' model library

    PCL839CS.LIB -' large' model library

PCL839CX.LIB denotes one of the above files. The decision of which model to use depends one your application, and especially memory usage. The relevant module has to be linked when you want to use the PCL-839 function calls.

PCL839.H

PCL839.H is a 'C'-language header file, and has to be included in your application source-file.

Usage : #include "PCL839.H"

SYNTAX. DOC

SYNTAX.DOC is a documentation file that contains descriptions of all the commands supported by PROG839.EXE, the interpreter program.