USB to Serial

From: support@ni.com
Subject: **Re: (Reference#7155691) Phone Support E-Mail**
Date: May 10, 2007 9:45:40 AM PDT
To: bsb@caltech.edu
Reply-To: support@ni.com
12 Attachments, 81.5 KB Save Slideshow

Occasionally it is convenient to use the Parallel Port for simple digital I/O. This document addresses how this might be done using VISA and register level programming. This document also addresses common mistakes, error messages, and problems encountered. This document does not deal with handshaking or PC to PC transfers, and only addresses specifics of the IEEE-1284 specification and communication when necessary. Those who want to learn more about the IEEE-1284 specification should consult the Developer Zone Tutorial: IEEE 1284 - Updating the PC Parallel Port which addresses this in much greater detail.

DC08 - DCDF

DCE0 - DCF7

Table of Contents:

(Embedded
image moved
to file:
pic00041.gif
)

Introduction

Normally the parallel port is used for output to a printer or other device. It sends data 8-bits or one byte at a time in parallel. The other lines available on the DB-25 connector are a combination of status lines, control lines, and ground lines. The status and control lines are used for handshaking, commands, and feedback when we are talking to a printer. We will have to concern ourselves with some of these lines in order to use the parallel port for our own purposes. In a Microsoft Windows environment, it is possible to get limited functionality from the parallel port using the same API that is used for serial communication. This means that in a Windows environment we can output data to the parallel port using the same VISA VIs that we would use for normal serial communication. However, the Windows API does not have built in support for input operations. Even though in some cases the hardware supports it, the software does not. This does not mean it is impossible to use the parallel port for input in LabVIEW, however, it does mean that it is not possible to use VISA VIs for input.

The parallel port on most computers uses the DB-25 connector pictured below in Figure 1. Table 1 shows the pin functionality.

(Embedded image moved to file: pic18467.gif)

Figure 1. DB-25 Connector - Facing Connector on Computer Back.

Note: The pin numbering is from 1-13 on the top row and 14 - 25 on the bottom going from right to left.

(Embedded image moved to file: pic06334.gif)

Table 1. Pin Functionality Diagram.

(Embedded image moved to file: pic26500.gif)

Parallel Port Configuration

Method
Using the parallel port for digital output is really rather trivial if working with the Windows API. The main trick is to tie pins 11(Busy) and 12 (Paper Error) to ground. Otherwise, the hardware driver will think the printer it is talking to is busy or experiencing an error and will not output any data. The port will maintain the last value written to it until another value is written or until the computer is powered down. Remember that in LabVIEW all serial communication needs to be sent as a string. Generally we will want to send 8-bit numbers to the port. This will require flattening the data to string so that the binary representation of the data does not change. We can use the Type Cast VI for the purpose.

(Embedded image moved to file: pic19169.gif)

Figure 2. Using the Type Cast function in LabVIEW.

If you output more than one byte at a time the driver will send them to the port in sequence and toggle the Strobe line (line 1) off and on for each byte. The timing involved varies from one computer to the next, but there are some standards in place. For more information about the timing characteristics, refer to the Developer Zone Tutorial: IEEE 1284 - Updating the PC Parallel Port

Refer to this Developer Zone Example: Using VISA to Access the Parallel

Port in LabVIEW for a LabVIEW program illustrating how to write to a Parallel Port.

Common Errors

-1073807330 (BFFF003E) - VISA "Could not perform operation because of I/O error"
or
LabVIEW hangs with Serial Write compatibility VI

Generally, these problems mean that lines 11 and 12 have not been properly grounded. You should be able to jumper these lines to any of the ground lines (18-25). Make sure you have not accidentally grounded lines 2 and 3 by mistake.

Error 37 using Serial Compatibility VIs

Serial devices need to be properly listed in the LabVIEW.ini file to use the Serial Compatibility VIs. Because it is now recommended to use VISA, the necessary line is not present in the INI file by default. Generally, the line should appear as:

serialDevices="COM1; COM2; COM3; COM4; COM5; COM6; COM7; COM8; COM9; COM10; LPT1; LPT2;LPT3; LPT4;"

LabVIEW assigns port numbers in the order of the list beginning at 0, so this would make COM1 equal to port 0 and LPT1 equal to port 10.

Search the KnowledgeBase for more troubleshooting ideas.

(Embedded image moved to file: pic15724.gif)

Simple Parallel Port Input and Output

Method

An alternative to the VISA VIs is to write data directly to the parallel port's hardware registers. In LabVIEW we can access the hardware registers using the In Port.vi and Out Port.vi located in the Advanced - Port I/O pallette. Because we are not using the higher level drivers, we do not have to concern ourselves with grounding any of the status lines.

Note: Accessing hardware registers on Windows NT or 2000 requires kernel level drivers. Refer to KnowledgeBase 2Q1FC3K8: How Can I Access Hardware Registers or Implement In Port and Out Port on Windows NT/ 2000? How Can I Read or Write Physical Memory? for more information.

The first step is to identify the base address for the parallel port. The

base address can generally be found in the Windows Device Manager under the Resources for the parallel port.

For Windows 9x:

1. On your Desktop, right-click on My Computer and select Properties.
2. Click on the Device Manager tab and find LPT1 under Plug and Play BIOS.
3. After selecting LPT1, click the Properties button.
4. Next select the Resources tab and the address should then appear next to Input/Output Range.

For Windows NT/2000:

1. On your desktop, right-click on My Computer and select Properties.
2. Now select the Hardware tab and click the Device Manager button.
3. LPT1 is found under Ports (COM & LPT).
4. Double click on LPT1.
5. Next, select the Resources tab and the address should appear next to Input/Output Range.

(Embedded image moved to file: pic11478.gif)

Figure 3. Parallel Port (LPT1) Properties in the Windows Device Manager

The base address is generally hex address 278, 378, or 3BC. There are several registers associated with the parallel port but for purposes of simple output we only need to concern ourselves with the Data register.. This is the first register in the I/O range, and so is located at the base address. The 8 bits in this first register map directly to the data lines (2-9). All we need to do is use the Out Port.vi to write the desired values to the port as shown in Figure 4.

(Embedded image moved to file: pic29358.gif)

Figure 4. Using Out Port.vi.

Common Errors

"Capability not supported" error form In Port.vi or Out Port.vi

On Windows NT and 2000 machines, additional AccessHW drivers are required. Refer to the Developer Zone Example: Port and Memory Utilities for Windows. Also, make sure you use the In Port.vi and Out Port.vi that come with the AccessHW driver and not the ones from the vi.lib.

Refer to KnowledgeBase 0TL70ATL: Using the Parallel Port as an Input/Output Channel for more information about regarding parallel ports and LabWindows/CVI.

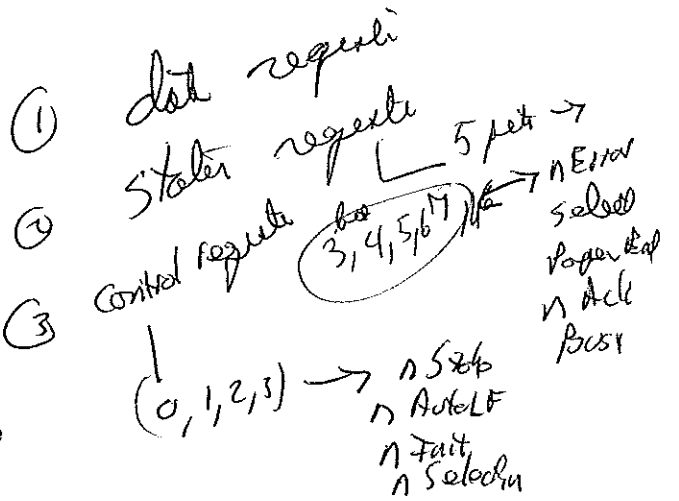
(Embed
ded
image
moved

to
file:
pic269
62.gif
)

Using the Data lines for Input

Method

As mentioned earlier, it is impractical to use VISA VIs to do input through the parallel port. But there are other issues which arise for input. For one thing it cannot generally be guaranteed that the 8 data lines can be made bidirectional. This issue is covered in more detail in the next section. For now lets assume we cannot use the 8 data lines for input. Fortunately, there are other lines available to us. The next register after the data register, base + 1, is the status register. Five of the bits in the status register map to lines on the 25-pin connector (Busy, nAck, PaperEnd, Select, nError). These map to bits 7,6,5,4,3 of the status register respectively as shown in Figure 5. The status lines are already configured for input. If we were communicating with a printer, these are the lines the printer would use for handshaking and feedback. The Busy line must be inverted because it uses inverse logic. If five or fewer lines are all that is necessary, we can simply use the In Port.vi to read in the status register. If we need more, however, we will have to start using control lines.



The control register is at address base+2. Much like the status register, several bits in this register map to lines on our connector (nStrobe, nAutoLF, nInit, nSelectIn). These map to lines 0,1,2,3 of the control register. These lines are normally output, but they can be configured for bidirectional signals. To turn them into inputs we simply have to set them all to logic high. If a line is set high and is externally grounded by a signal, the ground wins out and the bit reads low. We also have to be aware that the Strobe, nAutoLF, and nSelectIn lines are inverse logic. If we set the inverse lines low, we actually are setting them high at the connector which is exactly what we want. If we want to read one full byte, we can read the upper nibble of the status register and the lower nibble from the control register. This requires some binary manipulation in LabVIEW as shown in Figure 6, and we still have to be careful of the bits that are inverse logic.

(Embedded image moved to file: pic24464.gif)
Figure 5. Register Map.

Note: Number constants for digital logic are represented in binary.

(Embedded image moved to file: pic05705.gif)

Figure 6. Using the control and status lines for input.

Common Errors

"Capability not supported" error form In Port.vi or Out Port.vi
On Windows NT and 2000 machines, additional AccessHW drivers are required. Refer to the Developer Zone Example: Port and Memory

Utilities for Windows. Also, make sure you use the In Port.vi and Out Port.vi that come with the AccessHW driver and not the ones from the vi.lib.

Some parallel ports can be configured to use the data lines as inputs. It depends a great deal on the way the manufacturer designed the parallel port. With some models the data lines can be read the same way we read the control lines, by driving them to high logic so they will take on the value of an external signal. However, most parallel ports require that you set the direction bit for input. This is bit 5 in the Control register (base+2). If the port is capable of it, setting the direction bit high has the effect of making the lines tri-state so it can be driven externally. Sometimes it is also necessary to toggle bit 6 high or low. However, it should be noted that some manufacturers actually lock these bits so that software cannot change them. An example is shown below in Figure 7.

(Embedded image moved to file: pic28145.gif)
Figure 7. Using data lines for input.

To test whether your data lines can be used for input, try the following:

1. Set bit 5 of the control register high (at base address+2).
2. With nothing connected to the port, write a couple of values to the data port, and read each back after you write it.

If the reads DON'T match the writes, your port is probably bidirectional. Setting C5 disabled the data outputs and you're reading the open inputs of the data-port buffer.

If the reads DO match the writes, your port isn't bidirectional. The data outputs are still enabled, you're reading back what you wrote, and you won't be able to read external signals.

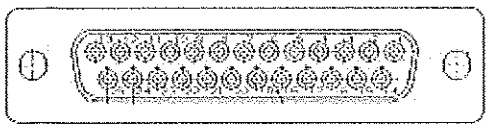
If it is possible to use your data lines for input, then you just need to set control register bit 5 high and read from the value of the data lines at the base address.

Common Errors

"Capability not supported" error form In Port.vi or Out Port.vi
On Windows NT and 2000 machines, additional AccessHW drivers are required. Refer to the Developer Zone Example: Port and Memory Utilities for Windows. Also, make sure you use the In Port.vi and Out Port.vi that come with the AccessHW driver and not the ones from the vi.lib.

Related Links:
Advanced Measurement Web Page
Developer Zone Tutorial: IEEE 1284 - Updating the PC Parallel Port

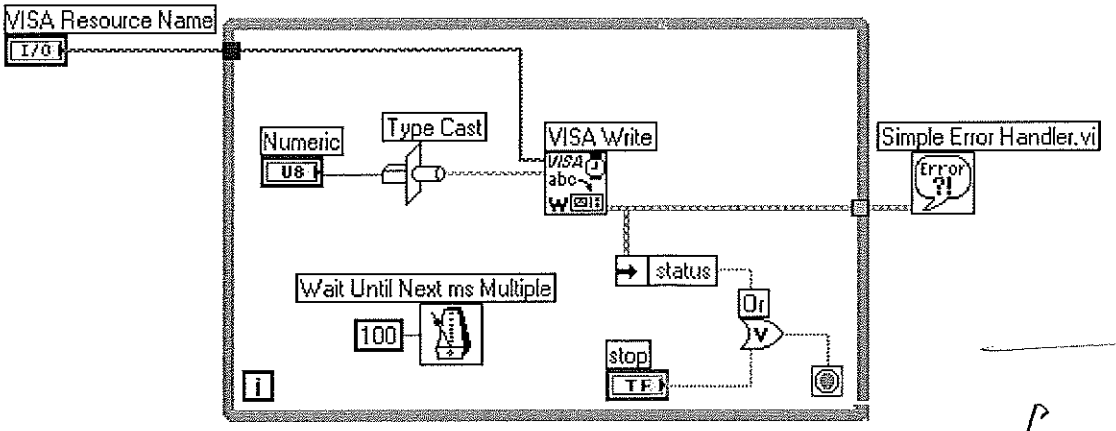
Note: Your reference number is included in the subject field of this message. It is very important not to remove or modify this reference

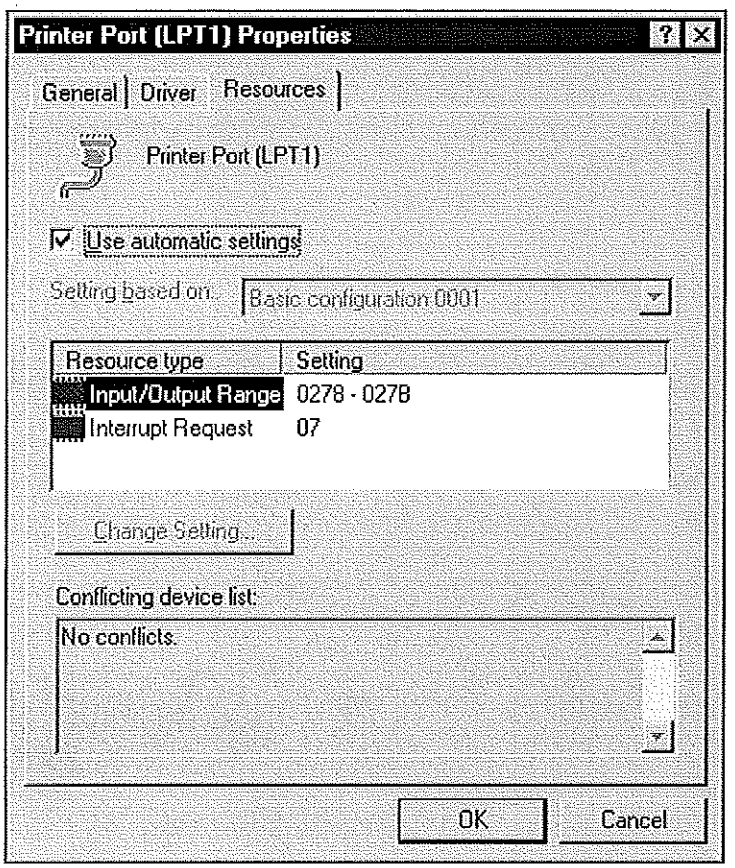


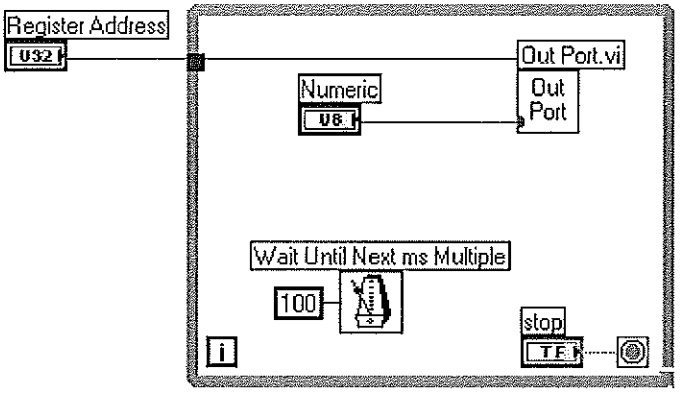
number, or your message may be returned to you.

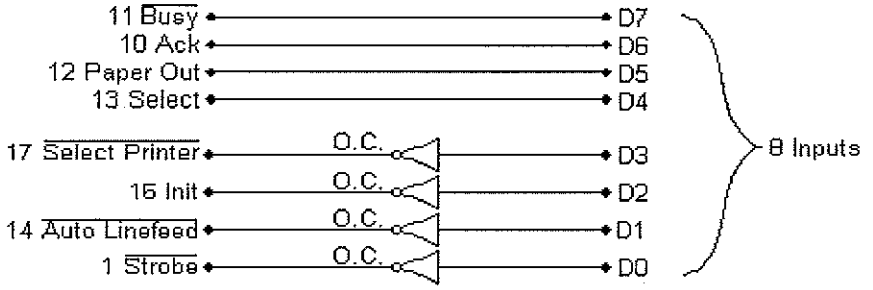
D-sub	Signal	Function	Source	Register	Register Bit #	Inverted at Connector?	Pin: Centronics
1	nStrobe	Strobe D0-D7	PC[1]	Control	0	Yes	1
2	D0	Data Bit 0	PC[2]	Data	0	No	2
3	D1	Data Bit 1	PC[2]	Data	1	No	3
4	D2	Data Bit 2	PC[2]	Data	2	No	4
5	D3	Data Bit 3	PC[2]	Data	3	No	5
6	D4	Data Bit 4	PC[2]	Data	4	No	6
7	D5	Data Bit 5	PC[2]	Data	5	No	7
8	D6	Data Bit 6	PC[2]	Data	6	No	8
9	D7	Data Bit 7	PC[2]	Data	7	No	9
10	nAck	Acknowledge	Printer	Status	6	No	10
11	Busy	Printer busy	Printer	Status	7	Yes	11
12	PaperEnd	Paper end, Empty	Printer	Status	5	No	12
13	Select	Printer selected (online)	Printer	Status	4	No	13
14	nAutoLF	Generate automatic line feeds	PC[1]	Control	1	Yes	14
15	nError (nFault)	Error	Printer	Status	3	No	32
16	nInit	Initialize Printer	PC[1]	Control	2	No	31
17	nSelectIN	Select printer (Place online)	PC[1]	Control	3	Yes	36
18	Gnd	Ground Return for nStrobe, D0					19,20
19	Gnd	Ground Return for D1, D2					21,22
20	Gnd	Ground Return for D3, D4					23,24
21	Gnd	Ground Return for D5, D6					25,26
22	Gnd	Ground Return for D7, nAck					27,28
23	Gnd	Ground Return for nSelectIn					33
24	Gnd	Ground Return for Busy					29
25	Gnd	Ground Return for Init					30
	Chassis	Chassis Ground					17
	NC	No connection					15,18,34
	NC	Signal Ground					16
	NC	+V5	Printer				35

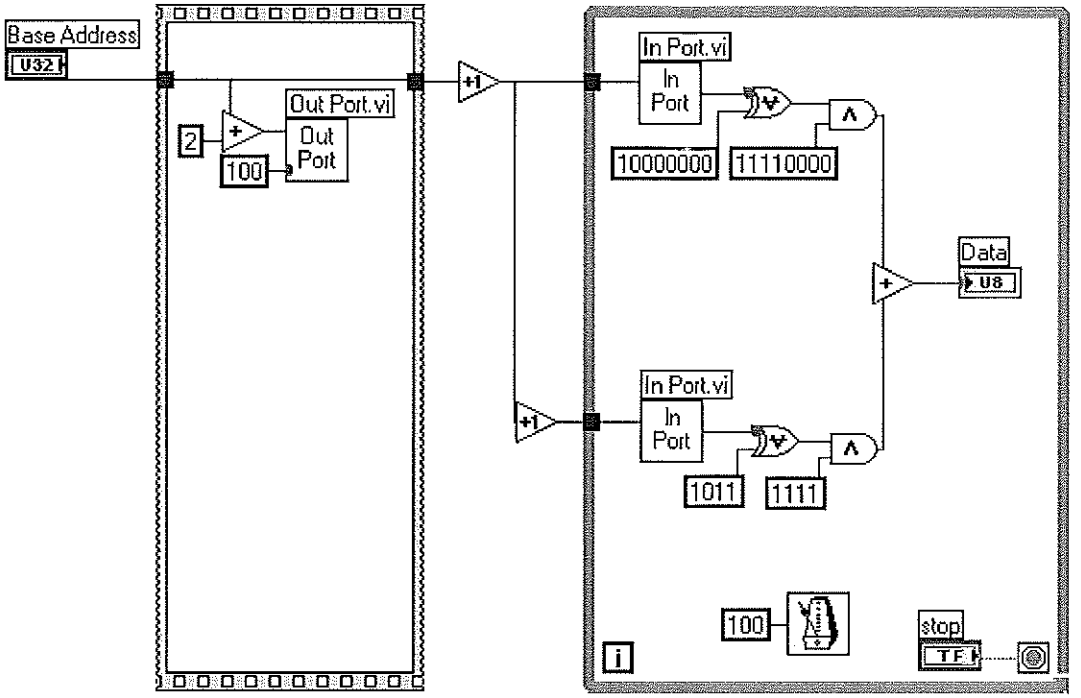
[1]Setting this bit high allows it to be used as an input [2]Some Data ports are bidirectional

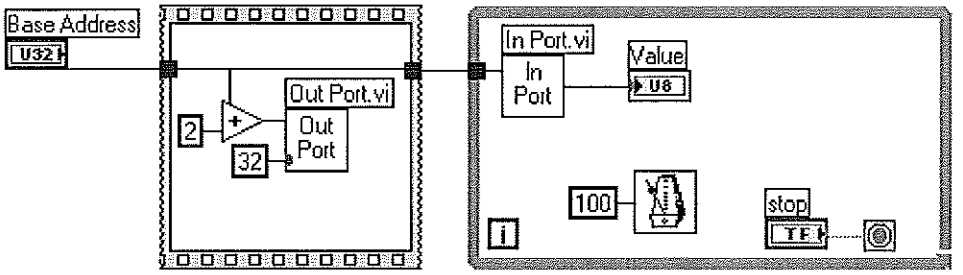



Printer Port (LPT1) Properties
General
Driver
Resources
Printer Port (LPT1)
Use automatic settings
Setting based on:
Basic configuration 0001
Resource type
Setting
Input/Output Range
0278 - 027B
Interrupt Request
07
Change Setting...
Conflicting device list:
No conflicts.
OK
Cancel


Register Address
U32
Out Port.vi
Numeric
U8
Out
Port
Wait Until Next ms Multiple
100
stop
TF
i


11 Busy
10 Ack
12 Paper Out
13 Select
17 Select Printer
16 Init
14 Auto Linefeed
1 Strobe
O.C.
O.C.
O.C.
O.C.
D7
D6
D5
D4
D3
D2
D1
D0
8 Inputs


Base Address
U32
Out Port.vi
Out Port
2
100
+1
In Port.vi
In Port
10000000
11110000
Data
U8
1011
1111
100
stop
TF
i


Base Address
U32
Out Port.vi
Out Port
2
32
In Port.vi
In Port
Value
U8
100
stop
TF
i