BiBaJa
Landscape Automation

Serial I/O User's Manual
2005-07-01A

INTRODUCTION

Bibaja's Serial I/O interface provides 4 inputs and 4 relay driver outputs for Mini-ITX and other PC's with standard serial port pin headers available on the motherboard. Each input may be used to sense contact closures at frequencies up to 100 closures per second. Outputs may be used to drive 12V relays for switching AC loads or to directly drive 12V devices. One pushbutton input and two LED outputs (red and green) are also provided.

Simple 9600bps (9600, 8 data bits, no parity, 1 stop bit) serial communications are used to configure and control the Serial I/O interface. The state of the inputs may be polled or configured to send updates to the PC when an input changes. Character echo may also be turned on for interactive operation of the Serial I/O interface.

Spring-cage clamps allow easy connect/disconnect of wires from outside of the PC case. Simply press the orange lever with a flat head screwdriver, insert the wire, and release. No screws to turn, no case to open.

Installation is a snap. Open the PC, insert the Serial I/O interface into a free PCI card slot space, screw down the bracket, and attach the power and serial cables. Use any free hard disk or floppy power connector to power the Serial I/O. Connect the serial header using the 10-pin IDC cable (provided).

Applications

Mini-ITX PCs are finding their way into many projects that require an embedded PC. Often, these projects require digital I/O interfaces between the PC and the outside world. The Serial I/O interface provides 4 protected inputs and 4 open collector relay driver outputs to provide an interface from the Mini-ITX PC to the outside world.

This section describes a few example applications utilizing a Mini-ITX PC and the Serial I/O interface. Log on to our support forum to exchange ideas with other members of the Bibaja community at http://www.bibaja.com/forum.

Desktop CNC Milling Machines

Linux users using Enhanced Machine Controller (EMC) software (http://www.linuxcnc.org) and other flexible CNC software will find the Serial I/O interface inputs useful for limit switches, and the outputs useful for controlling spindle motors, blowers, mist, flood, and more.

Three of the inputs could be used for the X, Y, and Z home limit switches using normally open micro switches.

Outputs can be used to drive relays to control AC loads such as the spindle motor, blower, mist, and flood.

Point of Sale Systems

Point of sale systems may integrate the Serial I/O interface for opening the cash drawer using a solenoid attached to one of the outputs, and sensing the state (open/closed) of the cash drawer using a micro switch coupled to one of the inputs.

Additional operator convenience buttons or a key switch input may be added. Operator convenience buttons such as a service call or register available button may be added to trigger the point of sale software to activate a lamp. The key switch would provide a simple way to log out of the register. Turn the key switch and take the key and the point of sale software will log the clerk out of the register. Insert the key and turn the switch and enter the password to log back in.

Outputs could be used to control the lamp for the register open/closed display. When the user logs out of the register, the lamp would turn off. Additional lamps such as a service call or register available lamp could be added and integrated into the point of sale software.

Security Systems

Security systems can use the 4 contact closure inputs to monitor door open/closed sensors, occupancy sensors, window open/closed sensors, beam break detectors, and any other contact closure switch type device. Outputs are useful for controlling lights, sounding alarms, and for controlling door latch solenoids to unlock doors.

12V supply and ground pins are provided to interface with devices such as PIR motion detector/occupancy sensors. Slimline Pet Immune PIR Motion Detectors, available from SmartHome (http://www.smarthome.com/7481A.HTML) easily attach to the Serial I/O interface. Security software running on your Mini-ITX PC can trigger lamps and sound an alarm when the security system is armed.

Driveway motion detectors may be integrated to detect cars pulling into the driveway, delivery vehicles, or detect when you come home late at night and trigger the pathway light. Sensors such as the Reporter Motion Detector available from www.smarthome.com (http://www.smarthome.com/7317.html) may be used to detect a vehicle or person walking in the driveway and sound an alarm, log an event in the security software, or light the pathway to welcome you home.

Thermostats include a switch to trigger your heating or cooling system. By adding a thermostat to an input of your Serial I/O, you can monitor and trigger an alarm when the temperature is too high or too low using a simple mechanical thermostat or an electronic thermostat. Lux manufacturers a low cost switch typically used to control baseboard heat. The Lux thermostat and others are readily available from your local hardware store.

Moisture sensors featuring a contact closure output may be tied to one of the inputs on the Serial I/O interface. If this moisture sensor is used to monitor a pipe, a contact closure signals a leak, and one of the outputs from the Serial I/O interface may be triggered to close an electronic master valve for the house, preventing expensive water damage from a broken water pipe or a valve that fails to close on a washing machine.

Magnetic reed switches are available for sensing when a window or door is open. Monitor when the front door opens and closes remotely through the internet and see when the kids come home from school, or detect when a thief enters your home. Card access readers could be added to your PC to automatically disarm the system so the alarm doesn’t sound when authorized people enter your home.

Outputs may be used to drive 12V solenoids or relays. Automatic door latches include a 12V solenoid to allow the door to open. When a card reader detects an authorized user, or the correct PIN is entered on a keypad, the door latch could be automatically opened by the PC controlling the home.

Relays attached to the Serial I/O output could be triggered when alarm conditions are present. Relays may be used to drive audible bell alarms, lights, and trigger contact closures to alarm systems by ADT and other monitoring service providers to contact local law enforcement.

The outputs are capable of directly driving 12V lamps. Red and green 12V lamps could be used to indicate whether the system is armed or disarmed, or in normal or alarm state.

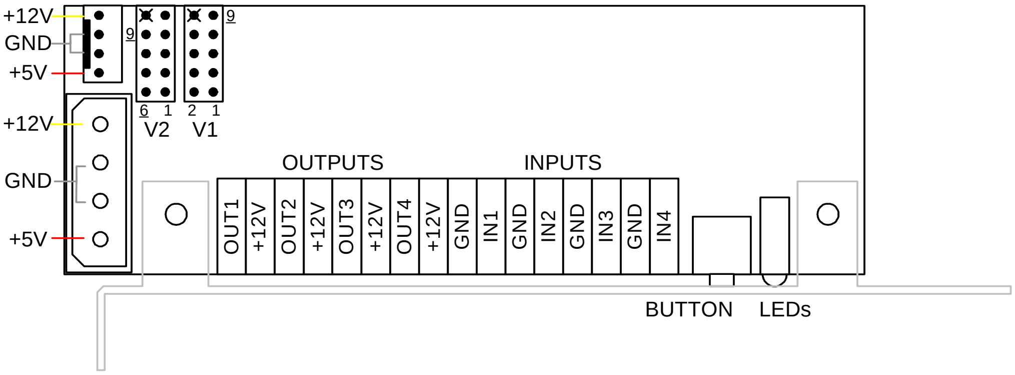
HARDWARE SPECIFICATION

Connectors

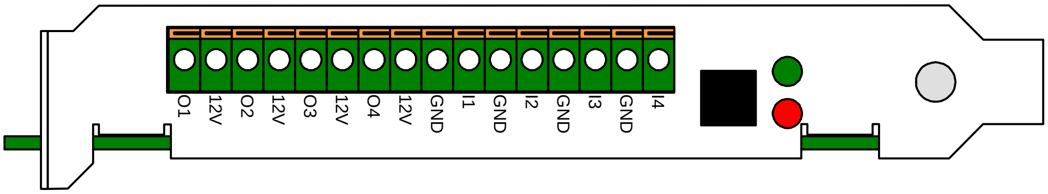
The Serial I/O interface provides the following connectors:

- 2 power connectors using floppy or hard drive style power cables
- 2 serial connectors (V1 and V2)
- 16 spring cage clamp connectors for outputs, inputs, +12V and ground

This PCB diagram shows the location of the connectors on the Serial I/O interface PCB:



Serial I/O device viewed from the PCI card bracket:



Floppy Power Connector

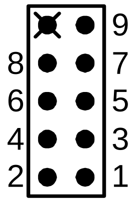
The floppy power connector can be used to power the Serial I/O interface board using a spare floppy power cable. The floppy power connector is located in the upper left corner in the PCB diagram above.

Hard Drive Power Connector

The larger hard drive power connector is located in the lower left corner of the PCB in the diagram above.

Serial Connector V1

Serial connector V1 is used to connect to VIA Mini-ITX motherboard communication port headers. The pinout is:

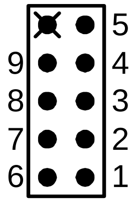


Signal	Pin	Pin	Signal
No Connect	X	9	RI
CTS	8	7	RTS
DSR	6	5	GND
DTR	4	3	RXD
TXD	2	1	DCD

NOTE: The only signals used by the Serial I/O interface are RXD, TXD, and GND.

Serial Connector V2

Serial connector V2 is used to make a ribbon cable between a 2x5 socket and a DB9 female connector. The pinout is:

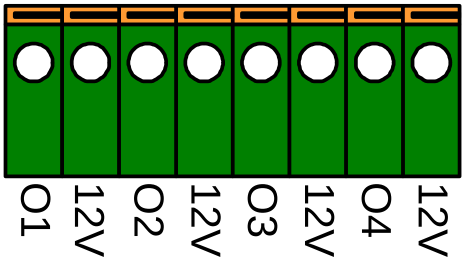


Signal	Pin	Pin	Signal
No Connect	X	5	GND
RI	9	4	DTR
CTS	8	3	SOUT
RTS	7	2	SIN
DSR	6	1	DCD

NOTE: The only signals used by the Serial I/O interface are RXD, TXD, and GND.

Output Spring Terminals (O1, O2, O3, O4)

Output spring terminals are paired with 12V supply terminals for convenience. All outputs are open collector meaning that they sink current only. To connect a relay with a 12V coil, the coil must be connected between a 12V terminal and one of the output terminals O1-O4.

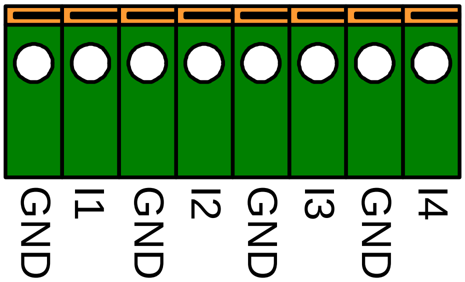


Current from the 12V supply is limited to 500ma by an internal PTC fuse. The 12V supply terminals may be used for powering external devices (PIR motion sensors, etc.). The total current draw of all attached devices from the 12V supply must be less than 500ma.

To insert or remove a wire, use a small flat-head screwdriver and press in against the orange lever. While holding the lever, insert or remove a wire into the desired terminal. Release the orange lever when finished. If you inserted a wire, give the wire a tug to be certain it is properly clamped in the terminal.

Input Spring Terminals (I1, I2, I3, I4)

Input spring terminals are paired with ground terminals for convenience. All inputs are pulled to +5V by an internal pullup. When the input is shorted to ground (when an attached switch is on, for example), the Serial I/O interface firmware indicates a logic 1 on that input. When the input is open (not connected or an attached switch is off), the Serial I/O interface firmware indicates a logic 0 on that input.



To insert or remove a wire, use a small flat-head screwdriver and press in against the orange lever. While holding the lever, insert or remove a wire into the desired terminal. Release the orange lever when finished. If you inserted a wire, give the wire a tug to be certain it is properly clamped in the terminal.

Electrical Specifications and Operating Conditions

Parameter	**Min**	**Typ**	**Max**	**Units**
5V supply current	---	20	75	ma
+12V supply current	---	---	500	ma
Current Sink (O1-O4)	---	---	500	ma
Operating Temperature	-10	25	70	C
ESD Protection (I1-I4)	---	---	8	kV air discharge
Toggle Rate (I1-I4)	0	---	100	Hz
Wire Gauge	26	---	18	AWG

SOFTWARE SPECIFICATION

Serial Port Settings

The serial port settings to communicate with the Serial I/O interface are:

9600 bits per second
8 data bits
No parity
1 stop bit
No flow control
(9600-8-N-1)

If you have hardware flow control on, or you have software flow control enabled, the device may not work properly. Flow control is accomplished by waiting for the response after every command is issued.

The Serial I/O interface is capable of sending 240 4-byte messages per second to the PC.

Command Strings

Communication between the PC and the Serial I/O interface is through command strings. Command strings may be issued manually from a terminal program or via software control. The format of the command string is:

<COMMAND><DATA…><TERMINATOR>

The <COMMAND> byte is a printable ASCII character. The following table gives a summary of <COMMAND> bytes:

<COMMAND>	**Description**
E	Event driven inputs, <DATA> is '1' to enable, '0' to disable
F	Full duplex, Serial I/O interface will echo each transmit character
H	Half duplex, Serial I/O interface does not echo transmit characters
I	Read inputs
O	Set or read output state, <DATA> is ASCII encoded hexadecimal data using characters '0-9' and 'A-F'
V	Read the version string, currently VSIOA20041

<DATA> is a sequence of ASCII encoded hexadecimal data. To output the hexadecimal data 0x7F, for example, the user should send the output command 'O' followed by '7', 'F', and the <TERMINATOR>. The current version of the Serial I/O interface only supports 8 bits of data. Entering more data will cause the first data to be shifted out to the bit bucket. For example, the command string "OFFA5<TERMINATOR>" will result in the outputs being set to 0xA5. The leading 0xFF is dropped. This feature allows you to write your code for future expansion to 12, 16, 24, or even 32 outputs by padding with leading 0's for off or leading 1's for on.

The <TERMINATOR> character is any one of the following 3 characters:

<TERMINATOR>	HexValue	Keyboard	Description
<EOT>	0x04	Control-D	End of transmission
<LF>	0x0A	Control-J	Line feed
<CR>	0x0D	Control-M	Carriage return

The Serial I/O interface firmware learns what terminator to use based on the command strings sent. If your software uses the <EOT> character, for example, the Serial I/O interface will terminate all command responses with the <EOT> character.

Command Response

Responses to a command string is a string resembling the command with a preceding '>' character. The format is:

```
><COMMAND><DATA><TERMINATOR>
```

The '>' character is used to distinguish command responses from event driven input data. For example, if an input request command 'I' is issued, and the inputs are changing, the '>' character provides a way to distinguish the response to the 'I' command from the event driven input event. Here is the flow of data using <LF> as a terminator:

```
I<LF>        // Sent from PC
I01<LF>      // Event driven input event (input 1 changed to 1)
>I01<LF>     // Response to I<LF> command
I00<LF>      // Event driven input event (input 1 changed to 0)
```

Waiting for the '>' response prevents confusion in thinking the first I01<LF> was the response to the I command and prevents further commands from being issued from the PC until the Serial I/O has finished the pending 'I' command.

Command Reference

Event Driven 'E' (Defaults to 0, disabled)

Enable or disable event driven inputs. Passing a '1' for the <DATA> will enable event driven inputs. '0' will disable event driven inputs. Event driven inputs are disabled by default at power up.

When enabled, any change in the inputs or pushbutton will cause the Serial I/O interface to send an unsolicited input command string to the PC. This string will not have a preceding '>' character making input events easy to distinguish from command responses. Every time event driven inputs are enabled, the current state of the inputs will be immediately sent to the PC. The following example demonstrates this:

```
E1<LF>       // PC sends command string to enable event driven inputs
>E1<LF>      // Serial I/O acknowledges
```

```
I00<LF>        // Serial I/O tells the PC the current input state
< time passes>
I10<LF>        // User presses the pushbutton, Serial I/O tells the PC the state
```

When the event driven inputs are disabled, the Serial I/O interface will acknowledge and will not send any further updates:

```
E0<LF>         // PC sends command string to disable event driven inputs
>E0<LF>        // Serial I/O acknowledges
```

Full Duplex ‘F’

Enable full duplex (echo) by sending this command. Full duplex means the Serial I/O interface will echo back each character as it is received. This mode makes interactive operation simpler by allowing the user to see what is typed. Software driven operation will typically use half duplex mode (echo off) since this is easier to parse.

To enable full duplex:

```
F<LF>          // PC sends enable full duplex/echo on command
>F<LF>         // Serial I/O acknowledges
```

Half Duplex ‘H’ (Default to half duplex)

Half duplex operation is the default for the Serial I/O interface. Half duplex is when the Serial I/O does not echo characters back to the PC. This mode is simpler for software driven operation as the software does not need to filter out it’s own commands from the serial input data stream.

To enable half duplex (echo off):

```
H<LF>          // PC sends half duplex/echo off command
>H<LF>         // Serial I/O acknowledges
```

Input ‘I’

Input a data byte from I1-I4 and the pushbutton. When event driven input mode is enabled, the Serial I/O interface will send input status to the PC every time an input changes using the input command string. The input bit assignments are:

Bit	**Input**
7	Reserved, read 0
6	Reserved, read 0
5	Reserved, read 0
4	Pushbutton
3	I4
2	I3
1	I2
0	I1

To read the current input state:

```
        I<LF>           // PC sends command to read inputs
        >I01<LF>        // Serial I/O responds with current input state
```

In the example above, the Serial I/O responds showing input I1 is at the ground state.

When an input is shorted to ground by a pushbutton press or a contact closure, the state of the input is 1. When the input is open, or the pushbutton is released, the state of the input is 0.

Event driven inputs may occur at any time an input changes. The following example shows a pushbutton sequence (released, pushed, then released again):

```
        I00<LF>         // Serial I/O indicates no inputs are active
        I10<LF>         // Serial I/O indicates the pushbutton has been pressed
        I00<LF>         // Serial I/O indicates the pushbutton was released
```

Note the lack of the preceding '>' character. This indicates the input status was due to event driven I/O instead of an input read command.

Output 'O'

Output a data byte to OUT1-OUT4 and to the red and green LEDs. The bit assignments for the outputs are:

Bit	**Output**
7	Reserved, write 0
6	Reserved, write 0
5	Green LED
4	Red LED
3	OUT4
2	OUT3
1	OUT2
0	OUT1

Writing a 1 to a bit will turn the corresponding output on. Writing a zero will turn the output off. For example, to turn on the Green LED and output OUT1, send the following command string to the Serial I/O interface:

```
        O21<LF>                 // PC Sends this command string to the Serial I/O interface
        >O21<LF>                // Serial I/O responds with this
```

The output command may also be used to query the current state of the outputs. If you don't wish to keep track of the state of the outputs, the Serial I/O interface will do it for you. The following example shows how to query the Serial I/O output state using the ‘O’ command:

```
O<LF>               // PC Sends this command string to Serial I/O interface
>O18<LF>            // Serial I/O responds with this
```

In this example, the output response from the Serial I/O interface shows the Red LED is on and output OUT4 is on.

Version ‘V’

Read the version of the Serial I/O interface using this command. To read the version string:

```
V<LF>               // PC requests the current version
>VSIOA20041         // Serial I/O responds with the current version
```

The version string consists of the following sections:

>V	– Acknowledge this is the version command response
SIO	– Serial I/O Interface
A	– Model A
2004	– Year the firmware was developed
1	– Minor version number

The year and minor version number will be incremented when the firmware changes. Major changes to the hardware will result in the model number changing. Major changes are adding new I/Os or other features.

INSTALLATION

Required Items

The following items are required to install the Serial I/O interface in your Mini-ITX or regular PC:

- One free serial port pin header matching either the V1 or V2 pin assignments
- One free power supply cable (either floppy or hard disk style)
- The Serial I/O interface
- 10 conductor ribbon cable with 2x5 sockets on each end (included w/Serial I/O)
- One free PCI slot or an external box for mounting the Serial I/O interface
- Screwdriver(s)

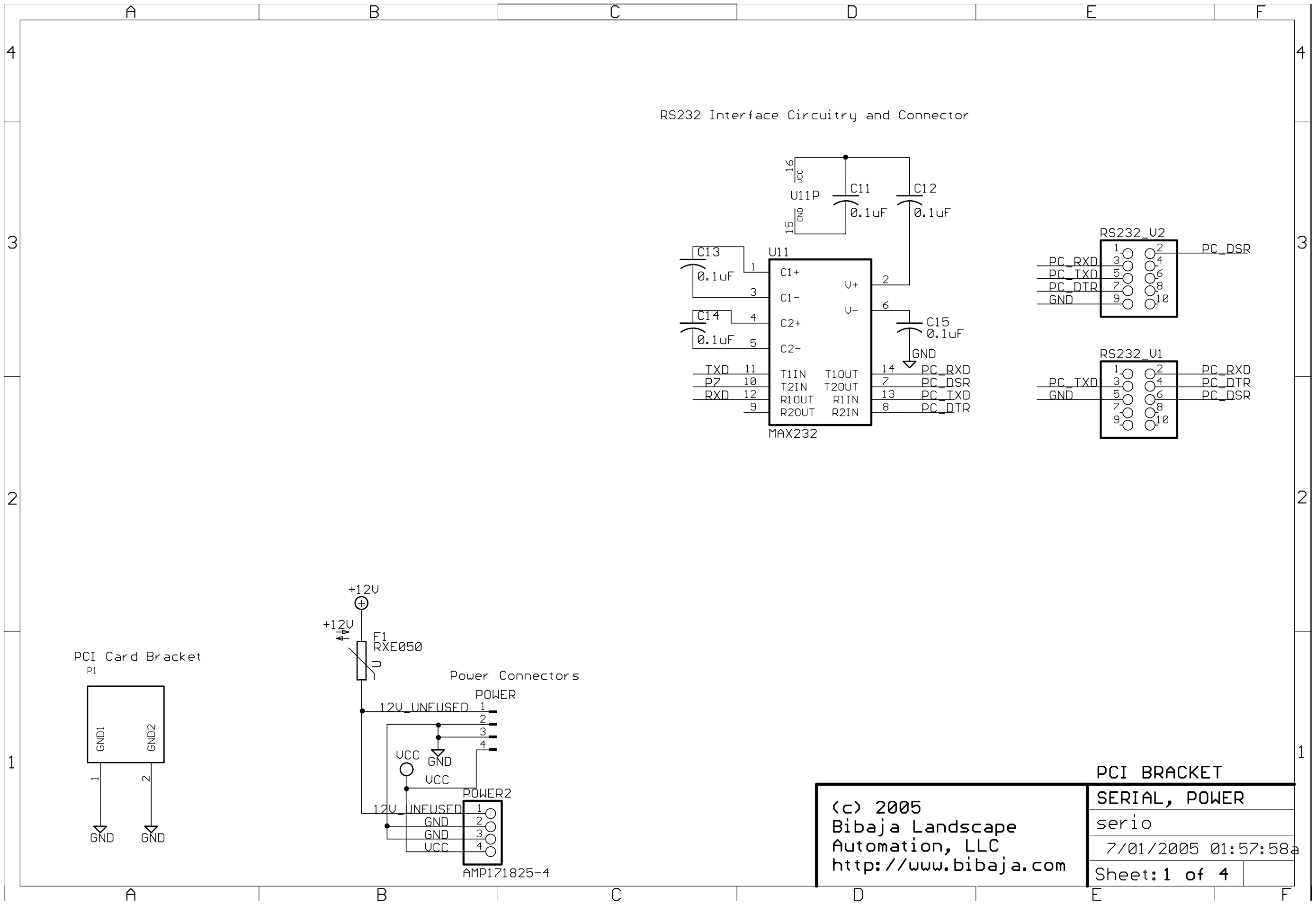
Preparing the PC and Installing the Serial I/O Hardware

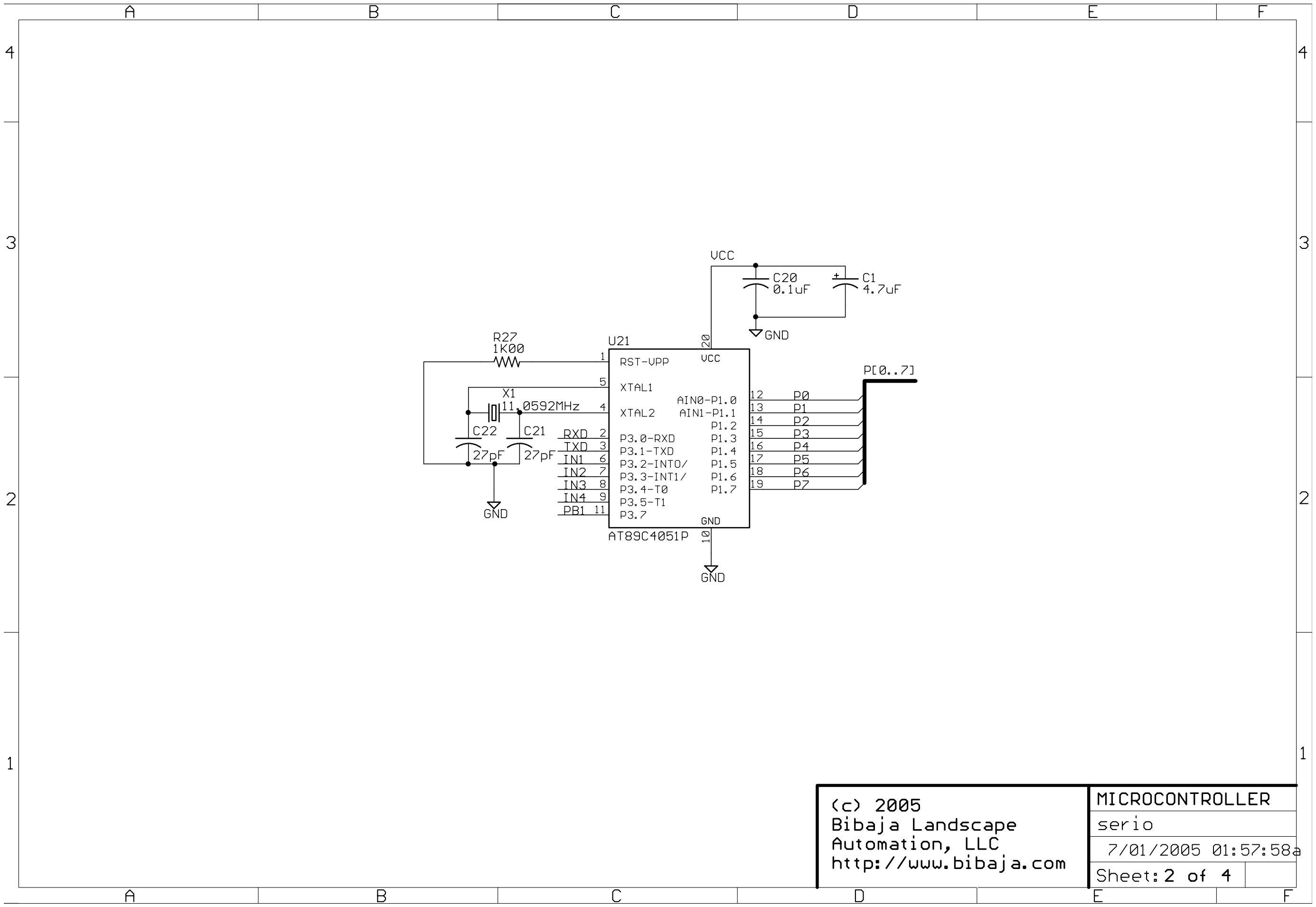
1) Power off the PC and use proper ESD precautions for following steps
2) Open the PC
3) Unscrew and remove the blank metal bracket from the target PCI card slot
4) Attach power to the Serial I/O interface
5) Attach the 10 conductor ribbon cable between the motherboard COM port header and either the V1 or V2 connector on the Serial I/O
6) Screw the Serial I/O interface into the PCI card slot
7) Power up the PC and test the Serial I/O interface

Testing the Serial I/O interface

1) Open a communications program, such as Minicom on Linux or HyperTerminal on Windows
2) Configure the serial port to 9600bps, 8 data bits, 1 stop bit, no parity, no flow control
3) Press return until you get the > character back acknowledging the return key was pressed.
4) Type V<LF> and check the version string response is “>VSIOA20041”
5) If you received the version string correctly, your serial I/O interface has been installed correctly.

APPENDIX A – SERIAL I/O SCHEMATIC





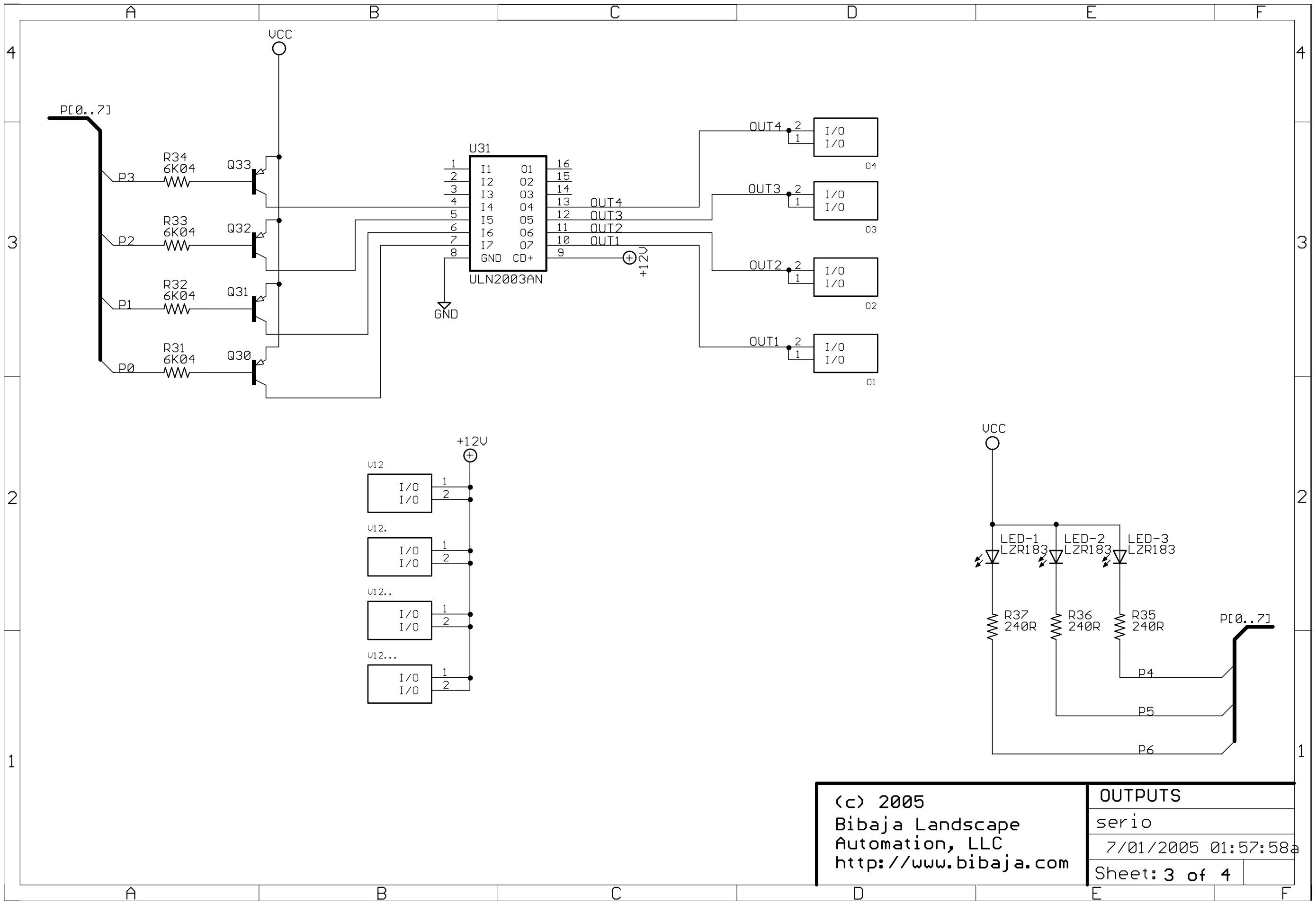
(c) 2005
Bibaja Landscape
Automation, LLC
http://www.bibaja.com

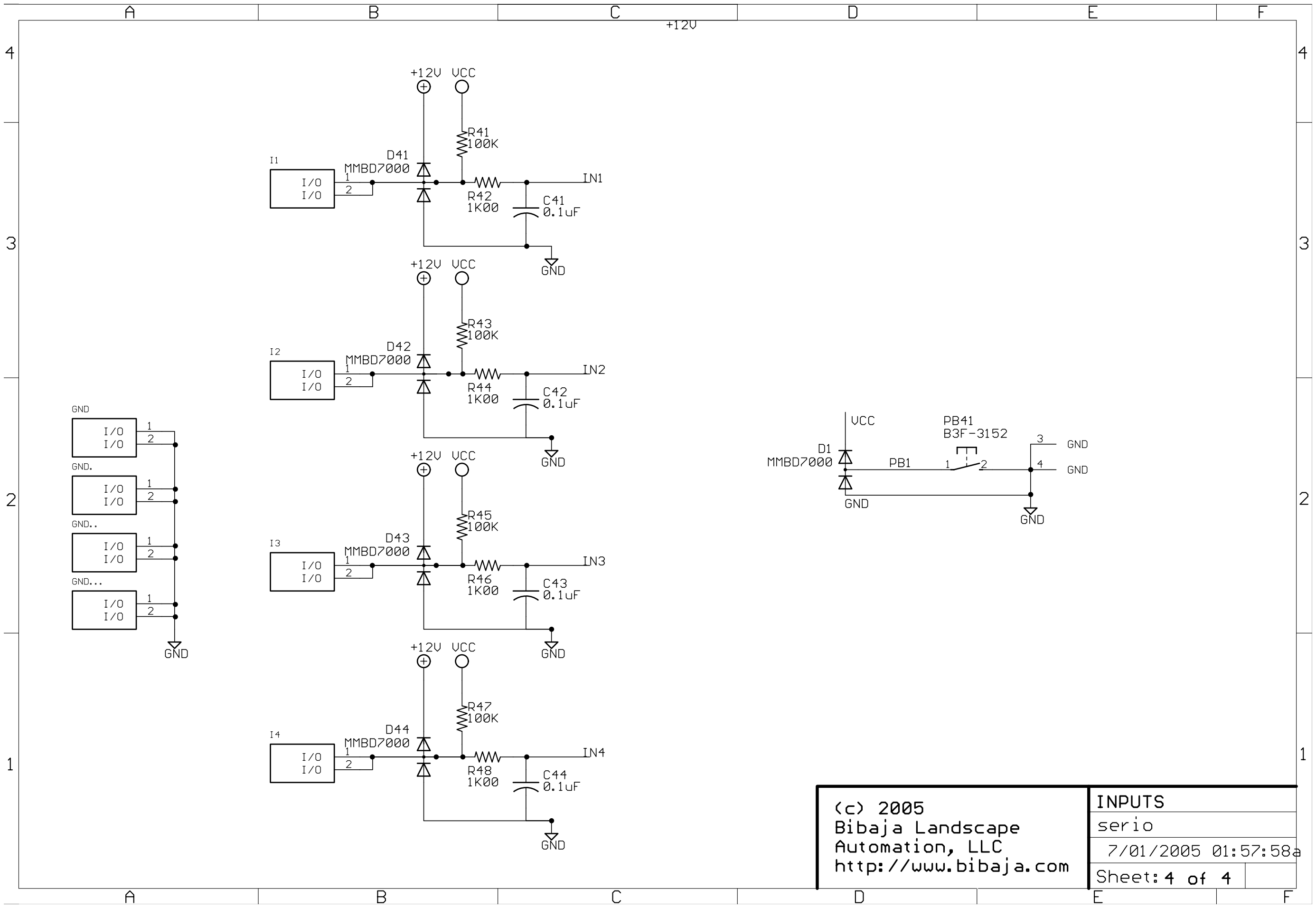
MICROCONTROLLER

serio

7/01/2005 01:57:58a

Sheet: 2 of 4


A
B
C
D
E
F
4
3
2
1
VCC
P[0..7]
R34
6K04
Q33
P3
R33
6K04
Q32
P2
R32
6K04
Q31
P1
R31
6K04
Q30
P0
U31
I1
I2
I3
I4
I5
I6
I7
GND
O1
O2
O3
O4
O5
O6
O7
CD+
ULN2003AN
GND
+12V
OUT4
OUT3
OUT2
OUT1
I/O
O4
O3
O2
O1
+12V
V12
V12.
V12..
V12...
VCC
LED-1
LZR183
LED-2
LZR183
LED-3
LZR183
R37
240R
R36
240R
R35
240R
P[0..7]
P4
P5
P6
(c) 2005
Bibaja Landscape
Automation, LLC
http://www.bibaja.com
OUTPUTS
serio
7/01/2005 01:57:58a
Sheet: 3 of 4


A
B
C
D
E
F
4
3
2
1
+12V
VCC
R41
100K
D41
MMBD7000
I1
I/O
I/O
1
2
R42
1K00
C41
0.1uF
IN1
GND
R43
100K
D42
MMBD7000
I2
R44
1K00
C42
0.1uF
IN2
R45
100K
D43
MMBD7000
I3
R46
1K00
C43
0.1uF
IN3
R47
100K
D44
MMBD7000
I4
R48
1K00
C44
0.1uF
IN4
GND.
GND..
GND...
PB41
B3F-3152
D1
MMBD7000
PB1
3
4
(c) 2005
Bibaja Landscape
Automation, LLC
http://www.bibaja.com
INPUTS
serio
7/01/2005 01:57:58a
Sheet: 4 of 4