DIY KIT 108. Serial Isolated Input/Output Module

Designed for control and sensing applications, this kit provides 8 relay outputs and 4 optically isolated inputs. It can be used in various applications including load switching, external switch input sensing, contact closure and external voltage sensing. It is controlled via a serial port using a set of simple text commands. After programming the PC can be disconnected from the PC without affecting the state of the relays. However, the kit is **not programmable in itself**. It requires an external controller such as a PC to send it commands to control the relays and monitor the inputs.

Connection to the isolated inputs and relay outputs is via "pluggable type" screw terminal blocks.

Using the serial port over a parallel port has several advantages:

1. Fewer wires are required (three instead of nine or more)
2. Serial cables can be a lot longer (up to 100ft), allowing "remote" control.
3. Serial ports can be used with any computer and operating system.

The kit is constructed on a double-sided, through hole plated printed circuit board (PCB) and fits in a plastic case measuring 140(W) x 110(D) x 35(H)mm. Screen-printed front and rear panels are supplied. The kit requires a 9-to-12V DC power supply. A wall adaptor rated at 500mA (minimum) is suitable.

SPECIFICATIONS

RELAYS	
Number of relays	8
Type	Form C, SPDT
Contact Capacity Resistive Load	15A 24VDC 15A 120VAC 10A 240VAC
Contact Capacity Inductive Load	5A 24VDC 5A 120VAC 5A 240VAC
Max. Allowable Voltage	AC 240V DC 110V
Max. Allowable Current	15A
Max. Allowable Power Force	1800VAC/250W
Min. Applicable Load	5VDC 10mA
Relay Life (Mechanical)	10 million operations
Relay Life (Electrical/Load dependent)	100,000 operations
Operating Time	10msec max.
ISOLATED INPUTS	
Number of inputs	4
Type	Polarised, opto-isolated (Not TTL/CMOS compatible)
Voltage	5 - 24VDC
Isolation (min.)	2500V peak 1775V RMS (1 sec.)

ASSEMBLY INSTRUCTIONS

Follow the component overlay on the PCB, starting with the resistors then diodes, IC sockets and crystal. Next fit the ceramic and monobloc capacitors, followed by the small electrolytic capacitors. Electrolytic capacitors are polarised, the positive lead is marked on the overlay, the negative is marked on the body of the capacitor. Leave the large electrolytic until later.

Fit the regulator next and use the 3mm screw and nut to attach the heatsink. Now insert and solder the 8 and 24 way "Dinkle" connector sockets. Make sure they are flush to the PCB before soldering. All that remains is to fit the DC jack and D9 connector, followed by the relays and finally the large electrolytic capacitor. Before soldering the DC jack cut the unused "side" tab flush with the PCB.

Do not insert any ICs yet. Connect a power supply to the DC jack (**centre positive**) and measure the regulator output (5V). If OK disconnect the power and insert the ICs. Take care that the ICs are the correct way around and none of the leads are bent under the body of the IC.

Before final assembly fit the reset switch to the rear panel. Connect the reset switch to the two pads marked S1on the PCB. Connect the power LED on the front panel to L1. We have provided L2, the data LED. We know that some people may not want to see a flashing light when data is flowing so if you want it drill a hole in the panel for it. The LED bezel is provided. Apply power again – the power LED should light. If not then the leads may need to be reversed. The data LED should be off. and the two LEDs to the front panel. The PCB uses only four of the eight mounting posts on the bottom of the plastic case. Remove the four inner posts. They can be easily "snipped" off using wire cutters.

Fit the front and rear panels to the PCB and hold in place. Slowly position the PCB into the base of the plastic case, making sure that the front and rear panels slide into the slots provided. Secure the PCB to the case using the self-tapping screws. Fit and secure the lid.

CIRCUIT DESCRIPTION

This kit started out as an extension of our popular Kit 74, "PC Printer Port Relay Board". The idea was to redesign the PCB to fit into a plastic case and add some inputs as well. At the same time a "latch" was added so that the PC could be switched off without affecting the relays. This proved to be difficult due to the PC's POST (Power On Self Test) function, which would operate and release relays during power up. The easiest solution was to use the serial port.

The circuit is very simple and straightforward. The brains of the kit is IC1, an 89C2051 microcontroller from Atmel. This was chosen because it has the required number of I/O pins and a built in serial port. It is pre-programmed to process all commands received via the serial port, control the relays and monitor the inputs.

IC8 provides conversion between TTL and RS232 signals. IC2 is an octal relay driver, ULN2803A, and is

used to drive each of the relays. The opto-couplers, IC4-7, are used to provide electrical isolation between the inputs and the rest of the circuitry.

At power-up, all relays are off (released) and the data LED is off. A reset switch is provided for manually turning off all relays. The data LED flashes whenever a valid command is received.

OPERATION

There are four ways to communicate with the kit:

1. A terminal emulator program running on your PC.
2. DOS type batch files.
3. Writing your own software.
4. Windows software downloaded from our website

TERMINAL EMULATOR PROGRAM

If you are running Windows then one is supplied with the operating system. In Windows 3.1 it is called "Terminal" and in Windows 95/98 it is called "HyperTerminal".

Of course, there are a number of third party terminal emulator programs that will work just as well. When using these programs the communication parameters need to be set for **9600 baud, 8 data bits, 1 stop bit, no parity and no flow control.**

Terminal for Windows 3.1

1. Go to "Accessories" group and double click on "Terminal"
2. Go to Settings → Communications
3. Select the following:
   - Baud Rate : 9600
   - Data Bits : 8
   - Stop Bits : 1
   - Parity : None
   - Flow Control : None
   - Connector : COM2:
     (or whatever COM port you are using)

The program is now ready to go. When exiting you will be asked "Do you want to save changes to the terminal settings?" Press "Yes". You will be asked to enter a file name. Type in a file name (eg. K108) and press OK. Your configuration settings are now saved.

From now on, run "Terminal" and go to File → Open. Select the configuration file you saved previously and click OK.

You can create an icon so that "Terminal" is run with those settings automatically.

- Select a Program Group then click on File → New → Program Item.
- In the "Program Item Properties" dialog box click on Browse.
- Set the File Name to "*.trm". A list of files with the ".trm" extension will appear. Select the file you saved previously (eg K108.trm) and press OK.
- You will return to the "Program Item Properties" dialog box. Type in a suitable description and press OK.

The new icon is now available. Double click on this icon to run the program.

HyperTerminal for Windows 95

1. Start → Programs → Accessories → HyperTerminal
2. Double click on "Hypertrm.exe" to start the program.
3. In the "New Connection" dialog box, type in a name for this connection (eg. K108). Select an icon for this new connection then press OK.
4. The "Phone Number" dialog box will appear. Go to "Connect using" and select "Direct to Com 1" (or whatever Com port the kit is connected to). Press OK to continue.
5. The "Port Settings" dialog box will appear. Select the following then press OK.
   - Bits per second : 9600
   - Data bits : 8
   - Parity : None
   - Stop bits : 1
   - Flow control : None

The program is now ready to go. When exiting you will be asked "Do you want to save session K108 ?" Press "Yes".

A configuration file called "K108.ht" is created with the communication settings you have selected. Its icon will appear in the "HyperTerminal" folder. Double click on this icon to start the program from now on.

HyperTerminal for Windows 98

Similar to W95. Run Terminal| File| Properties| and set the Com port. Still in 'Connect to' click Configure then set parameters. If after setting and saving (Save As K108.ht) you get funny characters on the screen then exit Terminal and re-start. Press Reset on the K108 panel. The # character should appear on the screen if the powered K108 is connected.

term.exe

We use our own terminal program term.exe to talk to Kit 108 from a PC. A full explanation how to get this program and set it up is given on page 6 below.

BATCH FILES

It is possible to control the kit using batch files. However batch files can only send data to the kit, they **cannot** process any data received back from the kit. Therefore it is possible to operate and release relays but **not** to read the status of the relays or the inputs.

If controlling relays is all you need to do then using batch files is a simple way to do it. Maybe get our Kit 74 which is specifically designed for batch control.
The following example shows the batch file commands required to send commands to the kit (assuming the kit is connected to COM2)

```
MODE COM2: BAUD=96 PARITY=N DATA=8 STOP=1
ECHO "N3" >COM2
```

The "MODE" command sets up the serial port to the required communication settings.

The "ECHO" command is normally used to write text to the screen. Using the output re-direction operator ">" causes the text to be sent to the serial port.

The second line sends the command "N3" to COM2 which causes relay 3 to turn on. Just keep adding ECHO commands to send further commands to the kit.

WRITING YOUR OWN SOFTWARE

As previously mentioned the kit is controlled via a set of simple text commands. Similarly all responses output by the kit are in text format. This makes it quite easy to communicate with the kit.

Using the serial port also makes it easier to write your own software to operate the kit. Visual Basic comes with a full set of functions to use with the serial port, as do most high level languages.

QBasic is available on most DOS based systems and is quite easy to use. The following QBasic example shows how to access the serial port and send commands to and receive data from the kit. The 'line numbers' are for explanation purposes only and are not required.

This sample program configures the serial port and reads the status of inputs 1-4. It then sets relays 1-4 accordingly. For example, if input 1 is 'high' and inputs 2-4 are not then relay 1 will be operated and all the other relays released. In other words relays 1-4 are set according to the condition of inputs 1-4.

```
1.  OPEN "COM2:9600,N,8,1" FOR RANDOM AS 1
2.  PRINT #1, "I0"
3.  INPUT #1, A$
4.  INPUT #1, A$
5.  PRINT "Input Status: "; A$
6.  PRINT #1, "R"; A$
7.  INPUT #1, A$
8.  PRINT #1, "S0"
9.  INPUT #1, A$
10. INPUT #1, A$
11. PRINT "Relay Status: "; A$
12. CLOSE #1
```

Line 1 configures COM2 for 9600 baud, 8 data bits, 1 stop bit and no parity. The COM port is assigned 'channel #1'. Change it to whatever COM port you wish to use.

Line 2 sends the command 'I0'. This command returns the status of ALL the inputs. Output is in ASCII HEX.

Line 3 reads a line of characters and assigns them to variable A$. Since the kit echos back all commands sent to it, this is simply reading back the previous command string, 'I0'. It is ignored.

Line 4 reads the status of the ALL inputs (the output of the 'I0' command). The input status is contained in variable "A$".

Line 5 prints the input status to the screen.

Line 6 sends the 'R' command letter followed by the input status contained in "A$". The 'R' command sets ALL relays directly according to the following hex byte. In this case the hex byte is the input status just read.

Line 7 reads in the 'R' command letter and hex byte that was echoed back by the kit, as in line 3.

Line 8 sends the command 'S0'. This command returns the status of ALL the relays.

Line 9 reads in the echoed command, as in lines 3 and 7.

Line 10 reads the status of the ALL relays (the output of the 'S0' command).

Line 11 prints the relay status byte to the screen. It should be the same as the input status byte.

Line 12 closes the COM port opened at the start of the program.

Note:

The module echoes all characters received back to the host computer. These characters must be processed or input overrun errors will occur. In QBasic , a statement such as INPUT (or LINE INPUT) will do that and should be used after each command is sent to the kit.

TESTING

The easiest way to test the kit is to use a terminal emulator program running on your computer. Run the program and set the communication parameters (as described above).

Connect the kit to the serial port on your PC using a **straight through 9 pin cable**. Switch on the power. The kit outputs an '#' character as a prompt to indicate it is waiting to receive commands.

Send a few commands to operate and release relays and check their status. Apply a voltage level to each of the inputs and read back its status. See also page 6 for this testing described in more detail.

COMMANDS

A set of simple text commands is used to control the relays, return their status or read the inputs. Each command consists of a string of ASCII characters followed by carriage return (Enter ↵).

The '#' character is output as a prompt to indicate the kit is waiting for a command. It should be on your screen. Each character received is echoed back. On completion of each command, good or bad, a carriage return/line feed combination is output followed by the '#' prompt. If the command or parameter is invalid, the command is ignored and a '?' is output before the next '#' prompt.

Note:

- Commands are not processed until the carriage return character is received.
- Commands can be in upper or lower case.
- Relays are numbered 1 to 8. Relay number '0' (zero) indicates ALL relays.

- Inputs are numbered 1 to 4. Input number '0' (zero) indicates ALL inputs.
- Where a hex byte is used, each bit within the byte indicates its corresponding relay or input. Bit 0 indicates relay or input 1, bit 1 indicates relay or input 2, etc.

Nx – Turn a relay ON (where x = relay number)

Eg. "N3" – turn on relay 3
"N0" – turn on ALL relays

Fx – Turn a relay OFF (where x = relay number)

Eg. "F3" – turn off relay 3
"F0" – turn off ALL relays

Tx – TOGGLE a relay on/off (where x = relay number)

Changes the state of a relay (ON to OFF, OFF to ON)

Eg. "T3" – toggle relay 3
"T0" – toggle ALL relays

Rhh – Set ALL relays directly

"hh" is a hexadecimal byte. Each bit within the byte indicates whether the corresponding relay is operated or not. If the bit is '1' then the relay is operated, if the bit is '0' then the relay is released.

Eg. "R55" – relays 1,3,5,7 ON, all others OFF
"R0F" – relays 1-4 ON, all others OFF

Sx – relay STATUS (where x = relay number)

A '0' (zero) is returned if the relay is released, '1' if operated.

The command "S0" returns the status of ALL relays. In this case a hex byte is returned. Each bit within the byte indicates the status of the corresponding relay.

Eg. "S3" – returns the status of relay 3
"S0" – returns the status of ALL relays

Ix – INPUT status (where x = input number)

A '1' is returned if the input is active or enabled, '0' otherwise.

The command "I0" returns the status of ALL inputs. As with the 'S' command, a hex byte is returned. Bits 0-3 indicate the status of inputs 1-4. Bits 4-7 are unused and are set to '0'.

Eg. "I1" – returns the status of input 1
"I0" – returns the status of ALL inputs

Note the output is in ASCII hex, not ASCII decimal.

A **special command, '?'**, will print the software revision date.

Windows Software. Try the new Windows software which can be d/l from:

http://www.crowcroft.net/kitsrus/diyk108.zip

This has been tested under W9x/NT/2000.

Please send me comments and error reports to

peterhk@kitsrus.com

IF IT DOES NOT WORK
Poor soldering ("dry joints") is the most common reason for the circuit not working. Check all soldered joints carefully under a good light. Re-solder any that look suspicious.

- Are all the components in their correct position on the PCB.
- Are the electrolytic capacitors the right way round?
- Are the ICs the right way around?
- Are any IC leads bent up under the IC body?
- Is the regulator output = 5V?
- Is it connected to the right serial port on your PC?
- Are you using a straight through cable?
- Is the serial port configured correctly?

Web Address & Email
You can email the developer of this kit at

frank2005@ozitronics.com

Information on other kits in the range is available from our Web page at:

http://kitsrus.com

Ask questions on our Kit Forum
http://www.beam.to/diyforum

Corrections
Version 2 PCB released november, 1999. Packing list corrected 1/2000.

5/2001. Software bug on Tx, x=2 thru 8, corrected in firmware.

DIY KIT 108. Serial Isolated Input/Output Module

PART LIST – KIT 108

Resistors (0.25W unless specified)

470R	R1,10	2
1K, 0.5W	R6,7,8,9	4
10K	R2,3,4,5,11	5

Capacitors

27pF ceramic	C11.12	2
100nF monobloc	C1,3,5	3
10uF 16V electro	C4,6,7,8,9,10	6
1000uF 25V electro	C2	1

Semiconductors

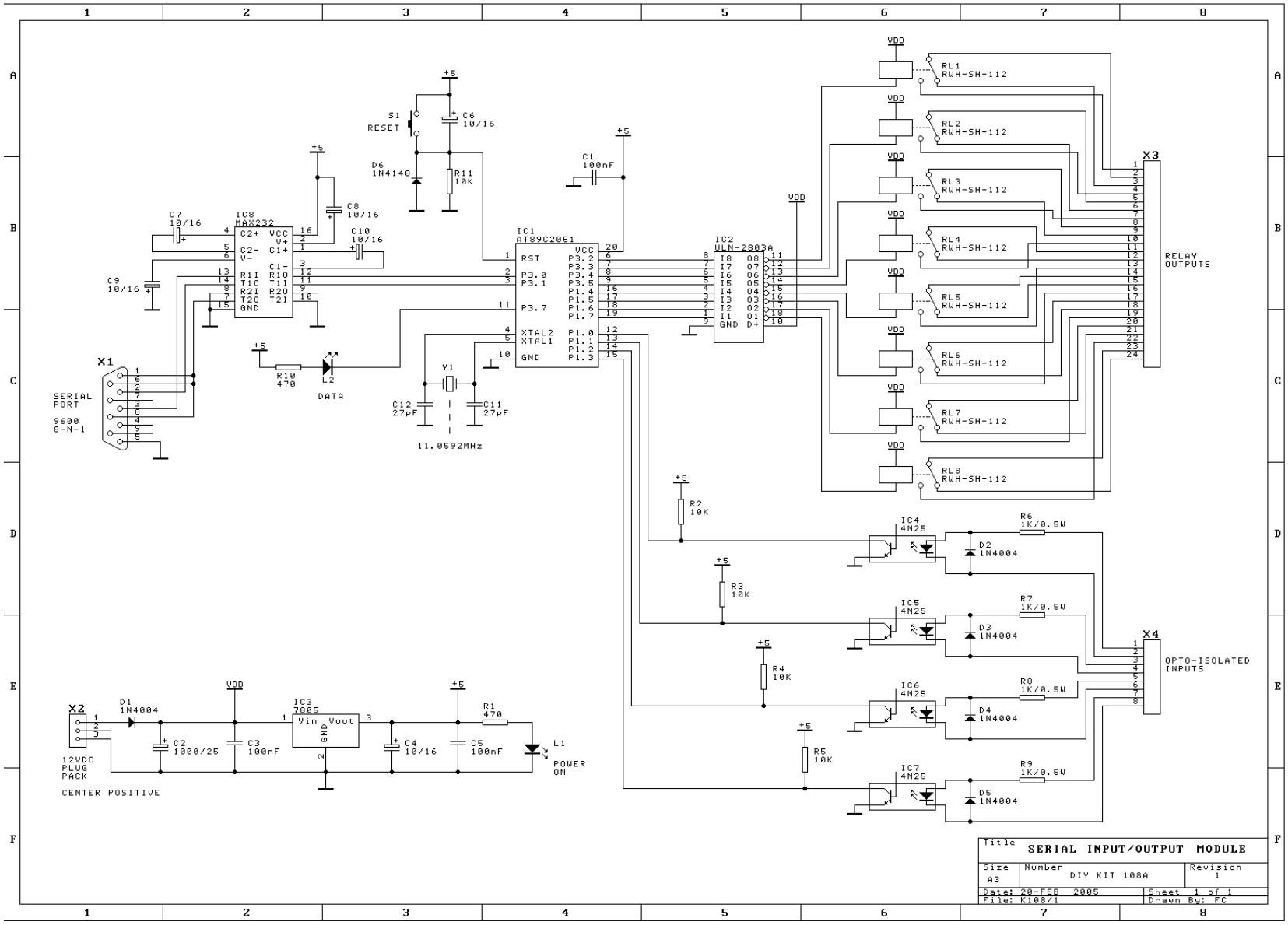
1N4004 diode	D1,2,3,4,5	5
1N4148	D6	1
4N25 opto-coupler IC	IC4,5,6,7	4
AT89C2051 uC	IC1 pre-programmed	1
ICL232 IC	IC8	1
RS232 driver/receiver		
ULN2803A	IC2	1
Octal open collector driver		
7805 regulator, TO-220	IC3	1
LED, panel mounting	L1,2	2
3mm red		

Miscellaneous

Crystal, 11.0592MHz	Y1	1
D9 connector	X1	1
PCB mounting, female		
Relay, SPDT	RL1,2,3,4,5,6,7,8	8
"Goodsky" RWH-SH-112D		
2.5mm DC jack	X2	1
Terminal socket	X4	1
8 way, PCB mtg, "Dinkle" 2EHDRC-08P		
Terminal socket	X3	1
24 way, PCB mtg, "Dinkle" 2EHDRC-24P		
Terminal plug, 3 way, to fit "X4"		8
"Dinkle" 2ESDV-03P		
Terminal plug, 2 way, to fit "X3"		4
"Dinkle" 2ESDV-02P		
Heatsink, to fit "IC3"		1
Pushbutton, panel mtg		1
IC socket, 6 pin, for "IC4,5,6,7"		4
IC socket, 16 pin, for "IC8"		1
IC socket, 18 pin, for "IC2"		1
IC socket, 20 pin, for "IC1"		1
Screw, 3 x 8mm, to fit heatsink to "IC3"		1
Nut, 3mm, to fit heatsink to "IC3"		1
Self tapping screws for mounting PCB		4
Plastic case, 140(W) x 110(D) x 35(H)mm		1
PCB, K108V2		1
Set of front & rear panels		1
Hookup wire, twin, 36cm (14")		

The source code for this kit is not available.

DIY KIT 108. Serial Isolated Input/Output Module



How to Use ‘term.exe’ Communications Program

Frank Crivelli (www.ozitronics.com) has written his own DOS-based comms (communications) program called ‘term.exe’. It is a simple, basic terminal program which does its job without a lot of ‘bells & whistles’. Also as of this moment he does not know how to write Windows software!

You may download it from **http://kitsrus.com/zip/term.zip**

How to install ‘term.exe’

This is a detailed explanation of how I have installed ‘term.exe’ in my Windows 98SE system. There are slight differences with Windows 95 and 2000 but it gives you an idea of how to do it.

1. Unzip ‘term.zip’.
2. Move ‘term.exe’ to a folder of your choice. I use the same folder as Hyperterminal. **C:\Program Files\Accessories\Hyperterminal**
3. Right click on some blank space on the desktop and select "New → Shortcut".
4. Click on the "Browse" button and find 'term.exe' on your hard disk in the above mentioned folder
5. Click on it and select "open". Click "Next", "Next" then "Finish".

You should now have an icon on your desktop. If you want to change the icon’s name then right click on it and select “Rename”.

6. Right click on this icon and select "Properties".
7. Click on the "Program" tab.
8. Go to the end of the "Cmd line" box and type in " 9600" (you must include the leading space). If you are using COM2 then type " /2 9600" instead. Notice that the “Working” directory/folder is set to the same as the “Cmd line”. You can change this if necessary.
9. Tick the “Close on exit” box. This will shutdown the DOS window when you quit ‘term.exe’.
10. Click on the “Change Icon…” button if you want to change the icon associated with this shortcut.
11. Click on the “Screen” tab and select “Full-screen”.
12. Click "OK".

To run ‘term.exe’ click on the desktop icon. Make sure you have 12VDC connected center positive. Press Enter. You should get a ‘#’. In my W98SE system I sometimes have to exit run it up to 2 more times in order to capture the serial port. You should not need more than three times.

Communicating with the Kit

See the COMMANDS section on page 3. Enter I0. You should get 00 in response. Enter N1. You should hear the faint click as Relay 1 engages. Press the hardware Reset button and you will hear it release. Enter N0 and all relays will turn on. Play with the other Commands as outlined on page 3.

Use a 2 position terminal block (as supplied) and connect between 5 – 24V as Inputs. Connect to Isolated Inputs position 1, +ve and –ve as shown on the end panel. Enter I0. You should get 01 as the status. In position 2 you should get 02, position 3, 04, position 4, 08. With no input you should get 00. With all inputs positive you should get 0F as the input status.

The power of the Kit lies in using software to read these inputs and act accordingly by setting or releasing relays.

- - - - - - - - - - - - - - - -