

## The IASI Interface and Protocol

### 1. Revisions

Rev	Change
Jan 2007	Draft

### 2. Background

The purpose of this document is to supplement the datasheets for the IASI products and give information about the protocol and help with writing software. The protocol uses simple text commands and once understood properly writing software to use the devices is a simple affair. There are two main parts to understanding the interface; the interface itself, being serial in nature and how the device responds to the commands given.

### 3. Introduction to IASI

The Intelligent Asynchronous Serial Interface (IASI) is a common standard that makes it much easier to control and use hardware from either a standard communication interface (terminal) or a microcontroller.

It is based on a very simple text command set and a flexible hardware interface. The 'Intelligent' aspect is derived from the fact that each particular IASI knows about the connected hardware so a simple command can make the hardware perform a reasonably complex function. Scroll text on an LCD display for example.

Another advantage is that on devices that require many i/o lines (24 on a standard LED dot matrix) this is greatly reduced by the use of a serial interface. Probably the biggest advantage of all is the ease at which something can be done. As an example to convert a voltage into a digital input, as a percentage, simply wire up the BV4106 and use the command VA. This of course can be done on most microcontrollers but each have their own way of doing things, just to set one up takes many hours of pouring through the data sheets.

When an IASI device is used in a microcontroller system this enables the controller and designer to concentrate on the important aspects of the design and control rather than the mundane job of controlling the hardware. It also means that the task of driving common peripherals is not being constantly re-invented.

#### 3.1. IASI Electrical Interface

An IASI device requires a power supply, transmit and receive lines as shown in table E1.

The interface is specifically designed so that it can be connected to either a standard com port (on a PC for example) or directly to a microcontroller UART or even a microcontroller port pin with a software generated UART (Universal Asynchronous Receiver and Transmitter). A five pin connector is used with normally only 3 or four pins being connected at any one time.

There are **TWO** receive lines, pin 1 receive line will accept normal 5V logic as presented by a microcontroller pin or UART and pin 4 will accept positive and negative voltages up to 15V that are normally present on a standard RS232 interface. Pin 4 will also invert the logic which is also normal for this interface.

The Baud rate is automatically detected at start up or it can be configured in software to a fixed, default baud rate. The device detects the baud rate on receiving a CR character (0Dh). Other received characters will be ignored until the Baud rate has been established.

The transmit pin has an open collector output<sup>1</sup> that has a pull-up resistor on board connected through a jumper. Where more than one device is used on the same serial line, only one jumper should be shorted. See the section on multiple devices for further information.

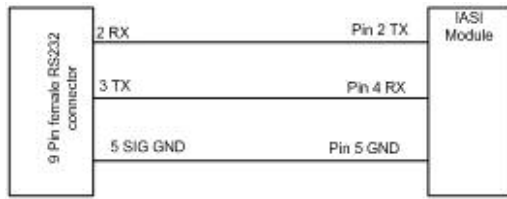
#### 3.2. Serial Connections

The device is designed to work in either of **two** modes: an INVERTED mode for connecting directly to an RS232 port (factory default) or a NON-INVERTED mode for connecting to a microcontroller UART.

As previously described there are two inputs, one for each alternative interface. On the transmit side (output from the interface) there is only one pin that takes care of inverted and non-inverted logic, this is configured in software. The output is 0 to +5V only, rather than the RS232 specification

<sup>1</sup> Not applicable to Bv4102, see section that deals with device specifics.

requiring positive and negative signals. On most RS232 specification interfaces this will work although it is not within the actual RS232 specification.



Pin	Name	Description
1	RX	Receive data in non-inverted form at +5V logic levels. Use this pin for connecting to MAX232 devices or directly to microcontrollers.
2	TX	Transmit (output) data. This is 0V and +5V, RS232 levels are not used. Devices will work without this connected but no feedback can be received. This pin is configurable in software to transmit either normal or inverted logic. (see multiple devices section 9.4)
3	+5V	Standard 5V power to the device
4	RX-Invert	Receive data (input) this will accept -12V to +15V volts in inverted logic as is normally available on a PC Com port. The format is RS232 1 start bit 8 data bits and 2 stop bits.
5	GND	Ground

**Table E1 Serial Connection Details**

**Figure 1 Typical Connection to a PC**

For more specific details on connecting to a PC serial port see section 8.

### 3.3. Factory Configuration

When an IASIM (Intelligent Asynchronous Serial Interface Module) leaves the factory it is configured to automatically detect the baud rate and interface with a standard PC com port. Regardless of the configuration the device will receive both forms of electrical interface (inverted and non-inverted) depending on which pin is used. So even if it may not be possible to see the output from the device, it will normally accept the input.

The transmit is configured by software and this is configured to invert. The interface does not need this pin to be connected to receive commands.

Factory settings can be restored both in software and hardware. The command ZF (see later) will restore the factory configuration. To restore factory defaults using hardware see Appendix A

## 4. The IASI Firmware

When writing software for an IASI device it is important to understand how the device reacts to commands given. The flowchart (Figure 2) illustrates how the IASO device responds to the commands and also shows how different configuration options effect how it behaves.

At start up the device will immediately run a macro, programmed by the user if this option is on, a peculiarity of this feature is that a user factory reset will turn off the macro facility but when the device leaves the factory the macro is on.

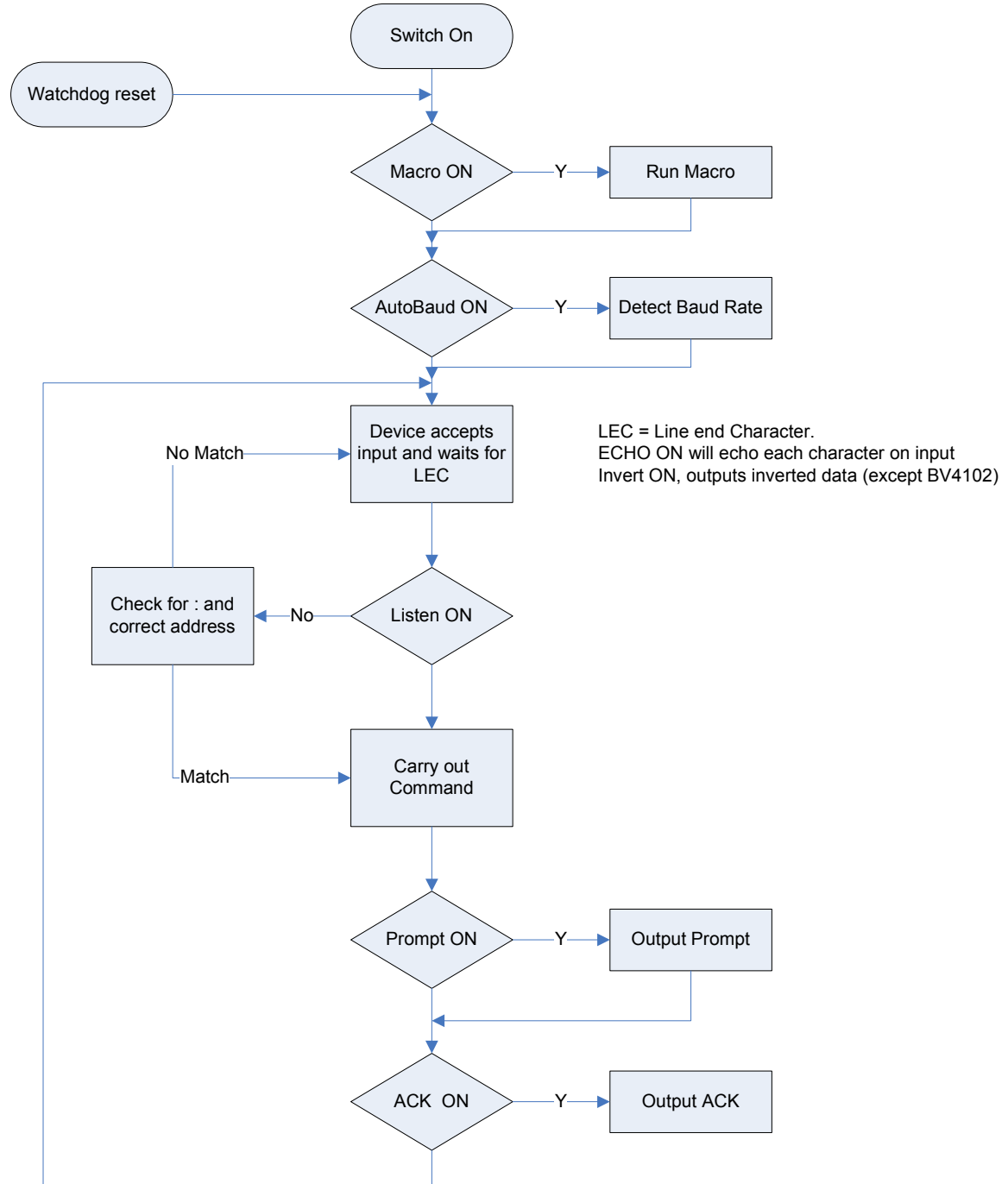
Once the macro has run the device waits in the autobaud section to receive an ASCII character 13 (0x0d), this is the enter key (Carriage return CR) on the keyboard. It will determine the Baud rate on detection of at least two CR values being received. Again this feature can be switched off and the Baud rate set to a fixed value by the ZB command.

The main command loop will accept any characters from the RX line(s) and put them into a serial buffer that is approximately 60 characters big, this varies from device to device. The device will continue to accept characters until a Line End Character (LEC) is received. If the buffer becomes full before the receipt of the LEC, and error is generated and the buffer cleared, any characters in the buffer will be lost. As may be appreciated the LEC is an important part of the command reception and by default is set to ASCII 13 (Carriage Return). This can be user configured to any character in the range 1..255.

One an LEC character has been received the firmware will attempt to interpret the text in the buffer. If 'Listen' is set to OFF then the command must be followed by a : and an address, if the device for example has an address of AA then :AA must be entered in order to progress further if AA is not received it will go back to accepting characters. If listen is ON then commands will be accepted without needing an address. The address is intended for multiple devices on the same bus.

The command is carried out next, whatever that may be. At the end of the command optionally a Prompt and / or and Acknowledge (ACK) character is output. The intension of the prompt is to give the user some feedback when configuring the device and the ACK is for microcontroller use, being an easily detectable single character.

Under some circumstances, where several devices are connected to the same bus, it may be possible to 'overwhelm' a particular device or there may be power problems that cause it to malfunction. If this happens there is a watchdog timer that will reset the device.



**Figure 2 Firmware flowchart**

Most of the control for how the above firmware operates is contained within 7 flags that can be configured with the ZC command.

## 5. Commands

The IASI works by issuing two letter commands, there is a common set of commands applicable to all IASI devices and a specific set of commands that apply only to that particular device. The common commands always begin with a Z, these are the Z commands.

The command protocol used essentially operates in two modes. An interactive mode and an addressing mode. The general format of the command is:

#### **Interactive**

<command><command-parameters><CR or ,>

e.g. **ZA**

#### **Addressing**

:<address><command><command-parameters><CR or ,>

e.g. **:00ZA**

The purpose of the interactive mode is to enable the user to learn the device capabilities, configure the device and also to be used in a system where only one device is connected to the bus.

This is the default protocol the prompt for this is:

**L>**

The 'L' indicates that the device is in 'Listen' mode and will react to any command given and also produce error messages if non-existent commands are given.

An alternative protocol is available that allows multiple devices to be connected to the same serial bus. This is configured using the **ZC** command (see later).

In non-listen mode the prompt is:

**:>**

This indicates that a colon and address is required. In this mode the device will only accept commands with its own address and ignore everything else. In this mode the previous **ZA** command would be:

**:00ZA**

Assuming that the device address is 00.

Note that all addresses must be preceded by a colon. The prompt can be turned off by the **ZC** command.

### **5.1. Addressing**

When not in Listen mode all devices have a two **character** address, this enables several devices to be connected to the same serial lines.

The address can be any two ASCII characters and it is case sensitive. Typical examples are **L1 AA 25**, etc. note that **aa** will be a different address to **AA**. The total number of address combinations is therefore 65025.

There is a special address **00** (zero, zero) that all devices will listen to regardless of their own address. There is nothing to stop more than one device having the same address, this can be useful if both devices always require the same commands.

As mentioned earlier if the device has been set to interactive mode (listen ON) an address is not required.

### **5.2. Command Format**

The command is a two character string **ZA** for example and is **not** case sensitive, so **zc** would be the same as **ZC**. All commands have two characters and in general the first character is a class of command and the following letter is a subclass of that command. All commands beginning with **Z** are common to all devices and usually deal with configuration and communication.

Other two letter commands deal with the device direct and the same commands may mean different things for different devices.

### **5.3. Command Parameters**

The command parameters can be anything and are specific to that command. In most circumstances spaces are ignored and can be used freely. Spaces do however serve to separate parameters (space is the delimiter) where more than one is required. For example given the command below that is used on the LCD IASM

**L>WT3 "Hello"**

The '3' requires a space after it so that the number can be distinguished from the text. In general though spaces are ignored and can be used freely.

## 5.4. Command Line (Buffer)

All commands strings are buffered in a nominal 64 byte buffer (varies with device) and devices will store any text received on the bus. Any lines greater than 64 characters will be ignored and generate an error so it is important not to exceed this figure on an automatic system.

Providing the debug level is set to greater than zero (see **ZD** command) then this will be reported as an error.

When a Carriage Return (ASCII 0dh) is received (configurable) the text string is passed to the command processor. In non-listen mode only devices with a matching address will respond, two or more devices can have the same address if required.

The line end character is set at the factory to be 13 (0dh) and this can be changed using the **ZL** command. Some systems send only this others send a Line Feed (10, 0ah) and others send both, either way round. The IASI will only accept the configured value and ignore or skip over anything else.

A comma (,) can be used to allow more than one command in a single line.

This is especially useful in non-listen (address) mode as the device only needs addressing once:

**[:>:00ZAL2**

**[:>:L2ZD1 or [:>:00ZD1**

can be sent as:

**[:>:00ZAL2,ZD1**

**(prompt)**

**(command)**

The above will set the device address to L2 and the debug level to 1.

## 6. Common Command Set (Z)

Command sets consist of a two character command. The first character usually refers to the general type and the second later is specific to that command. The Z command set for example is the common command set for all devices. The Z commands configure the device interface and communications.

### 6.1. The Star \*

As a general rule most commands issued on their own will return their value in printable format for example **ZD** will return the current debug level 0,1 or 2. This will be 'printed' on the terminal screen.

In absolute terms the values returned are in fact the values 30h, 31h or 32h which correspond to the printable values 1,2 or 3 (the ASCII codes for these characters). When used with a microcontroller for example it may be more convenient to know the actual BINARY value. By using a \* this will return the actual value thus.

#### **ZD\***

The above will return the actual binary value 1,2 or 3 which will not be printable on the terminal screen. This may be particularly important for numbers greater than 9 as the printed version of 10 is in fact 1 (31h) and 0 (30h), two bytes. Whereas the actual value 10 is only one byte and much easier to handle.

NOTE This also applies to device specific commands but see the datasheet for that device to make sure.

A full description of each command is given after the summary.

### 6.2. Summary

Command	Description
ZA	Address
ZB	Baud Rate
ZC	Device configuration
ZD	Debug level
ZF	Factory reset
ZK	Ack character

ZL	Line end character
ZM	Record Macro
ZR	Reset
ZT	Test macro
ZV	Version
ZW	Wait

Note that any example given will assume that the device is in interactive mode (listen flag on) so no address is required to precede it.

### 6.3. ZA

Name: **Address**

Command Parameters: [**\*address**]

Typical use **:00ZA or ZA (listen on)**

This command will show, confirm or set the address. To see the address of the current device simply use the command without any parameters:

#### **ZA**

Will return the actual address of the device, this will be 00 after factory reset.

#### **ZAK1**

This will set a new address for the device, K1. Addresses consist of two characters (including numbers) and are case sensitive so setting the address to AA will be different from aa.

#### **:K1ZA\*K1**

This command is used for confirming that a device is on the bus and working okay. In the above if device K1 is on the bus it will respond with ACK (see the ZK command for ACK). The single ACK character is usually easier to detect by a microcontroller than the two byte address returned without the \* parameter.

### 6.4. ZB

Name: **Baud rate**

Command Parameters: [**\***][**rate**]

Typical use **:00ZB or ZB (listen on)**

At factory default the Baud rate is detected by receiving a carriage return character (CR value 13, 0dh) from the sending device. This command will set a fixed Baud rate according to the following table:

0. 9600
1. 14400
2. 19200
3. 38400
4. 57600
5. 115200

For the device to choose from this table rather than wait for autobaud, the autobaud flag must be set or off or (N) using the ZC command. Not all devices are capable of going to the higher Baud rates.

The command enables viewing and setting of any of the above values 0-5 that correspond to the Baud rate. As an example if ZB is set to 3 and Baud detect is off, the device will communicate at 38400 Baud when switched on. If ZB is set to 3 and Baud detect is off then the Baud rate will be determined by the first acceptable Carriage Return character received.

The Baud rate can also be reported in binary, in the above example:

#### **BR**

will return 3 (ASCII value 33h)

#### **BR\***

will return the value 3 which will not be a printable character, this is useful for microcontroller input, see the section on the star command extension.

## 6.5. ZC

Name: **Configure**

Command Parameters: **n [\*][-w aaaaaa][-r]**

Typical Use **:00ZC3 -r**

This is probably the most complex command but will alter the way the device behaves and is only likely to be seldom used. The command is used for setting 7 binary (on or off) flags to control how the device works, these are:

1. ACK
2. Macro
3. Baud detect
4. Invert
5. Listen
6. Prompt
7. Echo

The above flags can either be on or off (Y or N) and they have the following meaning:

### **ACK**

When the line end character (ZL command) is received by the device there is an option to send the ACK character. If this is set to on (Y) then the ACK character will be sent. The ZK command sets the value of this character.

### **Macro**

When the device is first powered on, optionally a set of commands can be run, see the ZM, ZT commands. If this flag is set to Y then the macro commands will be run at start up. A reset to factory defaults will set this flag to off (N).

### **Baud detect**

The device can operate at a fixed Baud rate or detect the host Baud rate at start up. When this is set to Y the device will wait for input from the controller Terminal to determine the Baud rate, see the ZB command)

### **Invert**

This only applies to the **transmit line** of the device. When set to Y all transmitted data will be inverted, this is suitable for connection to normal RS232 type interfaces, a PC com port for example. There are some exceptions to this on particular devices, see the datasheet for the device.

### **Listen**

If this is set to Y a command will be accepted without addressing, this is useful for single devices connected to the bus and saves having to prefix all commands with colon, address.

### **Prompt**

When set to Y will output on the transmit line a prompt. This is useful for experimenting with the device when using a standard terminal. There are two types of prompt depending on how the listen flag is set, see the section on addressing.

### **Echo**

When this is set to Y all characters received will be transmitted to the transmit line. It is not advisable to have this on when used with automated systems. It may also slow down the command reception process.

The flags are stored in EEPROM as a single byte at a fixed location and to make setting up the device easier and to allow, if required a quick change over from one configuration to another, 8 additional configuration 'slots' have been provided.

The slots are written to using the -w switch, so for example to change the configuration in slot 7 use:

### **ZC7 -w NNNNNNN**

This will NOT change the configuration of the device, just the configuration in slot 7. As this is stored in EEPROM, the configuration will remain after power down. To change the devices configuration a slot

must be chosen, this will move the byte from the particular configuration slot to the current slot. To do this the `-r` switch is used on the command. To set the current configuration to that of slot 7 use:

**ZC7 -r**

The device will now take on the configuration of slot 7. Both of the above commands will cause a device reset, this is necessary to reflect the new configuration condition. It is also possible to set configurations that stop (or make it very difficult) communication with the device. If this is the case then a hardware reset is required.

Using the ZC command without a switch will show the configuration of that slot:

**ZCn or ZCn\***

where n is a number between 0 and 8 using ZCn without a \* will produce a string of Y and N. As an example:

**ZC0** will produce **NNYYYYN** given the factory defaults.

**ZC0\*** will return a value 1Eh which will not be printable using a Terminal but will be more understandable to a microcontroller.

**NOTES**

The purpose of the 8 configuration slots are to enable quick changing of device configuration. In most cases one of the default configurations will suffice and so the user may not be concerned with the individual aspects of each flag.

Not all devices have 8 configuration slots.

A factory reset will overwrite the first 5 slots with the factory defaults.

The first 3 (0-2) configurations are intended for using on PC terminals that have an inverted logic. The next three are intended for use with a microcontroller output that does not have an inverted logic.

**The use of this command with `-r` or `-w` will cause the device to reset, similar to using the ZR command.**

**6.6. ZD**

Name: **Debug**

Command Parameters: **[\*][n]**

Typical Use **:00ZD0**

There are three debug levels available. level 0 will not report any errors at all. This is used where multiple devices are on the same bus and the error reporting may interrupt normal command flow.

Level 1 will report to the normal transmit line and level 2 will report to the device if this is possible, LCD displays for example. Note if the device is not capable of displaying the error then level 2 will be equivalent to level 0.

As in the previous commands ZD on its own will report to the terminal in ASCII form and ZD\* will report in binary form.

A new level can be set by following the command with the new level, e.g.

**ZD1** set debug to level 1.

**6.7. ZF**

Name: **Reset to Factory Defaults**

Command Parameters: **none**

Typical Use **ZF**

This will set the device to the factory default configuration, see section 10 for what this is.

Providing there is communication with the device this command can be useful for establishing a known configuration automatically. For example:

```
ZF
CR (carriage return)
CR
:00ZF
CR
CR
```

The above should set the factory defaults even if the device is in interactive mode or not. This can then be followed by a desired configuration, **ZC3 -r** for example. The only time this may not work is if

the autobaud flag is not set and there is an incompatibility of Baud rates between the sending and receiving device.

### 6.8. ZK

Name: **ACK (acknowledge)**

Command Parameters: **[\*][n]**

Typical Use **ZK6**

When a line end (CR – configurable see ZL) is sent to the device the contents of the buffer are interpreted for a command and then acted upon. This takes a finite time to complete. In an automated system a delay can be used to allow for this. However the delay must be set for the longest time to ensure that all commands can be dealt with.

Optionally (see the ZC command) an acknowledge (ACK) mechanism can be switched on. If this mechanism is switched on, when the command has completed an ACK will be sent to indicate to the controller that another line is ready to be accepted.

This forms a simple but reliable handshake method that can be used for sending text files to a device or for using automated devices.

The ACK character can be set or displayed using this command.

**ZK** displays the character represented as a decimal printable number, **ZK\*** will return the ACK character as a binary value and **ZK6** will set the ACK character to the factory default value of 6

### 6.9. ZL

Name: **Line end character**

Command Parameters: **[\*][n]**

Typical Use **ZL13**

The line end character (EOL) is important as it informs the IASI to interpret the contents of the buffer. Unfortunately this character is not in universal use, some systems send 13, other send 10 and others send a combination of both which can be either way round.

The ZL command allows the specification of the character that will mark the end of line. This also gives the opportunity for any special requirements, e.g. 0 can be specified.

Examples:

**ZL** returns the value as a decimal number in text form.

**ZL\*** returns the value in binary form (1 byte)

**ZL2** sets the end of line character to 2. **WARNING** if you do this you will not be able to communicate with the device unless you end the command line with <CTRL>B (02)

#### NOTE:

The system will ignore or skip over any received values that are below a value of 32 (20h) except the following characters:

Escape	17, 1bh (interpreted as EOL)
Back space	8, 8h
ZL character	13, 0dh (by default)

### 6.10. ZM

Name: **Macro**

Command Parameters: **none**

Typical Use **ZM <see below>**

A macro is a common computer term to allow a sequence of commands to be recorded and played back at a later time. This command allows just that. The playback is normally at start up and allows programming of logos etc. on appropriate devices.

The command is used on its own; enter the command **ZM** and the following prompt appears:

**175>**

The number 175 (differing devices may have a different number) in front of the prompt shows the space available in characters. This will reduce as commands are entered.

Any valid commands can be used but the syntax is not checked until run time, the macro sequence can be checked with ZT. To exit this command use **Escape**.

Macros cannot be edited, the only way is to enter them from the beginning again. They can of course be sent as a text file from a terminal.

Note that if the macro flag is set, see the ZC command, the commands will be run at start up so care should be taken what commands are entered.

The factory defaults will not change the commands you have entered but it will set the macro flag to 0 so the macro will not run at start up. A brand new device has the macro flag set to 1 so it may be worth while setting this to 0 before experimenting. See the ZC command on how to do this.

The macro can be tested issuing the ZT command.

### 6.11. ZR

Name: **Reset**

Command Parameters: **none**.

Typical use **ZR**

This is the reset command and will reset the hardware and put the communication mode as if it had just been switched on.

### 6.12. ZT

Name: **Test Macro**

Command Parameters: **none**

Typical Use **:00ZT**

This will test any stored macro and should be used before setting the macro flag.

### 6.13. ZV

Name: **Version information**

Command Parameters: **none**

Typical Use **ZV**

This simply returns a sting that contains the firmware and device version information.

Additional information is also displayed but this does not apply to all devices:

The ZV command is also used for detecting errors and indicating if the factory defaults have been set. The format is:

**[WnFn] <version information>**

Where n is either 0 or 1. Under normal circumstances this would read [W0F0]. For information about this see the separate headings under watchdog and factory reset.

NOTE that reading the data clears it so if the first read was:

**[W1F0] <version information>**

The second call of this command would read:

**[W0F0] <version information>**

### 6.14. ZW

Name: **Wait**

Command Parameters: **n (1-65534)**

Typical Use **ZW110**

This command will simply suspend the device for a length of time. The time is entered in approximately 10ms intervals, so 100 is 1000 ms which is 1 second.

A typical use is to 'hold back' a text file when downloading or in a macro so that animated displays or actions can be implemented. At any time during the delay if a character is received the delay will abort.

This command is not suitable for highly accurate delays as the 10ms interval is only approximate and may vary, particularly from device to device.

Example:

**ZW 200**

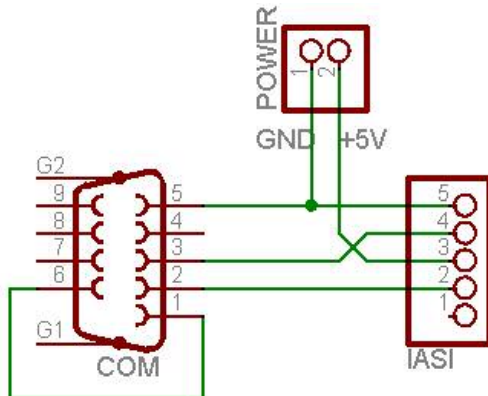
The above example will delay approximately 2 seconds.

## 7. Z Command Error Codes

Error codes will be displayed if the debug level (ZD) is set to greater than 0.

Code	Description
1	End of input buffer reached, buffer full. If this happens the whole of the input line will be ignored.
2	Unknown command, the command issued is not in the command table for this device.
4	Non-valid hex, where a command is expecting a hex value and something is entered outside 0-9, a-f, A-F. <b>NOTE</b> This is rarely used as most numbers values are in decimal.
5	Incorrect parameter following command. This refers to commands that expect a particular value to follow. This error will indicate that a command has been entered that is not in the correct / expected format.
6	Bad decimal number. This is because the command is expecting a number and something else has been entered. Note that a space should always follow a number.

## 8. Connecting and Configuration



**Figure 3 Connection wiring**

The above wiring diagram shows the connections to a standard PC 9 way com port (RS232 connector). Pin 1 of the IASI has no connection as this is used to connect to a microcontroller UART.

The factory defaults will work with the above configuration.

Start HyperTerminal or some other terminal software, BV Terminal is ideal and can be obtained from [www.byvac.com](http://www.byvac.com) The following settings should be used:

Baud rate 9600  
 Start bits 1  
 Stop bits 2  
 Handshake none  
 Local echo on

(The Baud rate is not that important as the IASI will adjust to the terminals Baud rate)

Power up the device and press 'return' a few times making sure that the terminal is active (mouse cursor in the terminal window). The device should respond with:

**L>**

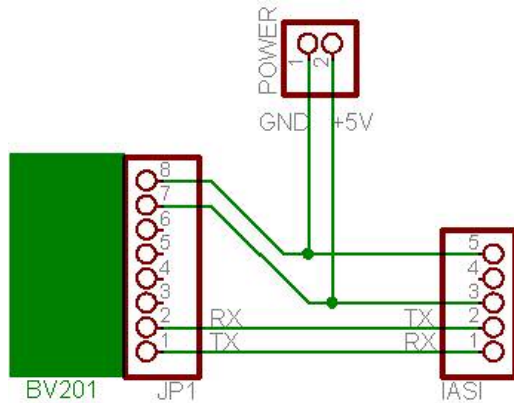
when it is ready to accept commands. If this is not the case check the power supply and terminal settings.

The device is now ready to accept commands, try **za**. This will return 00 which is the default device address.

After this prompt the device is now ready to be configured but it may be worthwhile spending some time looking at the command options available.

### 8.1. Reconfiguring

It may be that you want to use the device through a line driver device (MAX232) or microprocessor UART without bothering with the PC com port cable. This is also possible.



**Figure 4 Using non-inverted**

The above illustrates the connections used for, in this example a BV201 board that simply translates the PC com port to non-inverted 5V logic levels.

Using a BV101 a USB solution could be provided and there would be no need for a separate power supply.

See [www.byvac.co.uk](http://www.byvac.co.uk) for these products.

Start up HyperTerminal or BV Terminal as before and press enter two or three times. In this instance the terminal will not show the prompt as the output is the wrong polarity, simply ignore any characters returned and type:

**ZC3 -r**

Now press return two or three times to establish the Baud rate and the **L>** prompt should appear. This illustrates that it is not necessary to receive anything from the device in order to reconfigure it.

The command ZC3 -r, loads configuration 'slot' 3 into the device. See the ZC command for standard configurations.

## 9. Microcontroller Use

### 9.1. Simple

There are several ways the device can be used with a microcontroller. The simplest way is to use it in the default mode:

ZC0 -r (inverted)

ZC3 -r (non-inverted - most likely)

For an output device, and LCD for example no return value is necessary but sufficient time should be given for the command and device to operate. The TX line (from the IASI) does not even need to be connected. Note that although feedback is described below, this configuration can also use feedback by looking for the '>' of the L> prompt.

### 9.2. With feedback

A more sophisticated method would be to have some feedback from the device. There is also no need for a prompt.

The table shows a configuration that will do this for a single device. To place this is say, slot 6 requires the following command:

**ZC6 -w YNYNYNN**

To load this into the device type:

**ZC6 -r**

and issue two or three CR to establish the Baud rate.

<b>ACK</b>	<b>Y</b>
Macro	N
Baud-d	Y
Invert	N
Listen	Y
Prompt	N
Echo	N

As described in the ACK section of the ZC command the purpose of this is to tell the master controller (microcontroller) that another command can be issued. This can greatly speed up the transfer process.

**9.3. Baud rate**

It may be convenient to fix a Baud rate rather than having to establish it each time by sending CR. Two things need to be done for this:

ACK	Y
Macro	N
<b>Baud-d</b>	<b>N</b>
Invert	N
Listen	Y
Prompt	N
Echo	N

First set the desired baud rate according to the table illustrated in the ZB command. As an example, to set the baud rate to 38400 do this:

**ZB3**

Now configure the device so that the automatic Baud rate detection is switched off as in the table above.

The summary of commands would be as follows:

**ZB3****ZC6 -w YNYNYNN**

**CR** ( carriage return)

**CR**

**CR**

**ZC6 -r**

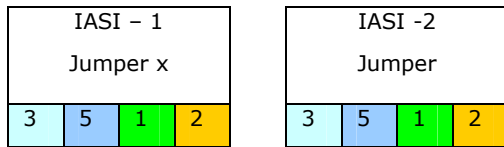
The device will retain the above configuration until either reconfigured or the factory defaults have been reset.

**9.4. Multiple Devices**

In the above examples the configuration assumed that only a single device was connected to the serial bus and so no addressing was required, commands such as ZA and ZL would be readily accepted by the device (called listen mode).

If more than one device is connected then, unless both are not required to respond to the same command, each device needs its own address.

Four (three if no feedback is required) connections are required. The diagram shows two devices but more devices can be added.



- 3 – Connect +5V together
- 5 – Connect Ground together
- 1 – Connect RX together
- 2 – Connect TX together

Note: the electrical interface is designed to accept multiple devices on the RX side for either input from RS232 (inverted) or UART (non-inverted). The TX side however will only transmit to multiple UART devices in non-invert mode.

**Jumper:** remove all of the jumpers except one

**ZAD1**

Will set the device to address D1 (note that d1, lower case would be different). It is also a good idea to stop error messages:

**ZD0**

Finally configure the device to stop listening and no prompt.

ACK	Y
Macro	N
Baud-d	N
Invert	N
<b>Listen</b>	<b>N</b>
<b>Prompt</b>	<b>N</b>
Echo	N

To illustrate this as a series of commands the following would be required. Using slot 7 to hold the configuration data.

```
ZAD1
ZD0
ZC7 -w ynnnnnn
CR (carriage return)
CR
CR
ZC7 -r
```

The device can now be commanded as follows:

**:D1za**

This will return 3 bytes 'D' '1' and 6 (if using the default ACK).

**10. Restoring Factory Defaults**

Factory defaults can be restored either by software or hardware. The factory default condition is:

- ZA00 ( address 00)
- ZC configuration 0 (see the ZC command)
- ZK6 ( ack character)
- ZL13 ( end of line character)
- ZB0 ( baud rate 9600 ) \*\*

\*\* Note that the Baud rate will be ignored as the default Baud detect, set by configuration slot 0 is set to on.

The first 6 configuration slots will be set as the table given in the ZC command. Any user information placed in those slots will be overwritten.

**Applicable only to some devices:**

Where a factory reset has occurred for whatever reason the ZV command will indicate this. A typical read out would be:

[W0F1] <version information>

The F1 in the square brackets indicates that a factory reset has taken place, this data is stored in EEPROM so even if the device is switched off the data still remains. The act of reading the data using the ZV command will clear the F1 back to F0, so this can only be seen once per factory reset.

**10.1. Software**

To set the above condition in software use the ZF command, once the ZF command has been issued the device will reset and wait for at least two CR (carriage returns) from the host device. This of course can only be used if it is possible to communicate with the device. If no communication is possible then a hardware reset will be required.

**10.2. Hardware**

It is unlikely that this will ever be used but has been provided in the unlikely event that a combination of configurations has rendered the device unable to communicate.

The technique is:

1. Power down
2. Connect shorting link
3. Power up
4. Power down
5. Remove shorting link

It will require a short piece of wire and the two points are normally at the ends of a row of 5 holes, one of which will be square. This may vary from device to device. As an example the BV4103 has been shown.



**Figure 5 Shorting link as on a BV4103**

**11. Watchdog**

\*\*\* Not applicable to all devices \*\*\*

Most devices have additional error protection in the form of a watchdog timer. The timer is set internally and is constantly kept up to date by the firmware. Should anything unforeseen go wrong and the firmware is unable to update the timer a reset will occur.

The most likely scenario for this happening is if the serial interface is overwhelmed by input that the built in overrun protection cannot cope with. Or some power glitches can cause the device to become 'confused'.

If a watchdog time out occurs, a flag is written to the EEPROM. This can be read using the ZV command:

[W1F0] <version information>

The output from the ZV command will indicate that a watchdog time out has occurred by a '1' following W in the example above. The act of reading the information will clear the flag, so the next read using ZV will show:

**[W0F0] <version information>**

This information can be used to debug the external driving software.

## 12. Writing Software for IASI

This section deals with the practicalities of using an IASI device whether this be on a PC running C or on a microcontroller system there are some basic principles that will help produce a better system.

Any problems that have been reported are usually part of a system that contains multiple devices. To produce a reliable system the timing needs to be understood in context with the IASI device.

<command><device interprets and takes action><device responds with prompt>  
|-----> time

The above diagram shows what happens in an IASI device over time, of course this can vary depending on the configuration used. The aspects to consider are:

1. All devices listen, even though it is not its address
2. It takes a finite time to process a command and can vary from command to command within the same device.
3. Devices will accept input even whilst carrying out a command, the text will go into the buffer
4. The buffer will continue to fill until a line end character is received.

A device will accept all text into the buffer regardless of what that text is. Once an End Of Line (EOL) character is received the device will interpret the text from the beginning of the text buffer. This includes the address interpretation, if the address is not for the device it ignores the input. The outcome of this is that the buffer will clear when the device receives an EOL and is able to interpret that line.

The following scenario can occur that will cause a device to behave erratically: Text has been sent to command device 1, whilst this is carrying out the command further text is sent to other devices. Device 1 will still continue to receive text into its buffer and will not be able to respond to the EOL until it finishes the command. When the command has finished it will then interpret the text in its buffer. It is possible that the buffer could overflow before this happens, generating an error. This error should not cause a problem, however if a command intended for the device is partly received (at the end of the buffer) when a buffer full is generated. The device will miss the command as the error will clear the buffer, thus causing erratic behaviour.

To reduce the risk of errors two things will help. The first thing is to only send an EOL character; by default the EOL is CR (Carriage Return). On most systems this is often followed by a LF (Line Feed), in C use "\r" rather than "\n". The second is to make sure that none of the devices on the bus are being swamped by text, remember that all devices interpret even though the command may not be meant for them. To do this it may be necessary to introduce delays between commands.

Strictly speaking each device should be dealt with in turn, waiting for the device to respond before sending the next command. This will completely eliminate any problems. On display (output type) devices it is possible to turn off all of the response mechanism from the device and run a system many times faster by exploiting the input buffer, the risk has been highlighted above.

A final note is that all of the devices incorporate a watchdog mechanism so that if an error does occur that completely confuses the device it will reset. If the device is built as part of a system then by proper use of configuration this should just cause a glitch rather than stop the device working.

Taking this a stage further, for the above reason it is not a good idea to put set up code in the main program because if the device does watchdog reset it will not be useable as it requires this start up code. As an example the default mode for an LCD BV4103 is normal mode, if the system requires extended mode then put this into the device macro so that it is in extended mode at start up. As a test, unplug the device from a running system and plug it back in again, if the code is resilient and the device set up correctly it will start to work as intended, most of the time. Most of the time because this is a very severe test and does not give the device sufficient time to reset properly before receiving commands.

## 13. Troubleshooting

**The device does not show an L>prompt anymore and does not appear to respond to any commands.**

*The IASI device has a very comprehensive set of input options and it is quite possible to set a combination of input options that are not compatible with what is connected. It will appear that the device has stopped working when in fact it is simply operating in a different mode.*

What's required is to get back to the factory default and take it from there. To do this follow the methods used in section **10 Restoring Factory Defaults**

Note that any factory macro which displays a start up sequence, the "ByVac" in the case of the LCD display for example will not be displayed. This is because the default setting is to turn the macro run off, this is a safety feature in case there is some erroneous code in the macro which prevents the display from starting.

**Under software control the device appears to be or non-working, but using it with a terminal it works fine.**

This is probably due to the EOL character. The device will only respond when an EOL is received. By default this is set to ASCII 13. Although the system will ignore character values below ASCII 32 on some commands these may cause problems LF before CR for example.

For the most reliable results make sure that the software only outputs a single EOL. In C for example terminating the line with `\n` (CRLF) may produce unreliable results, use `\r` instead.

As an alternative consider setting the EOL to something else e.g. `!` and output this when terminating a command e.g. `ZA!`.