



W.A.S.P.

Windows Analogue Signal Processor

**Installation and Users Manual
(Including WaspWare Software)**

**Available exclusively from
PC Control Ltd.
www.pc-control.co.uk**

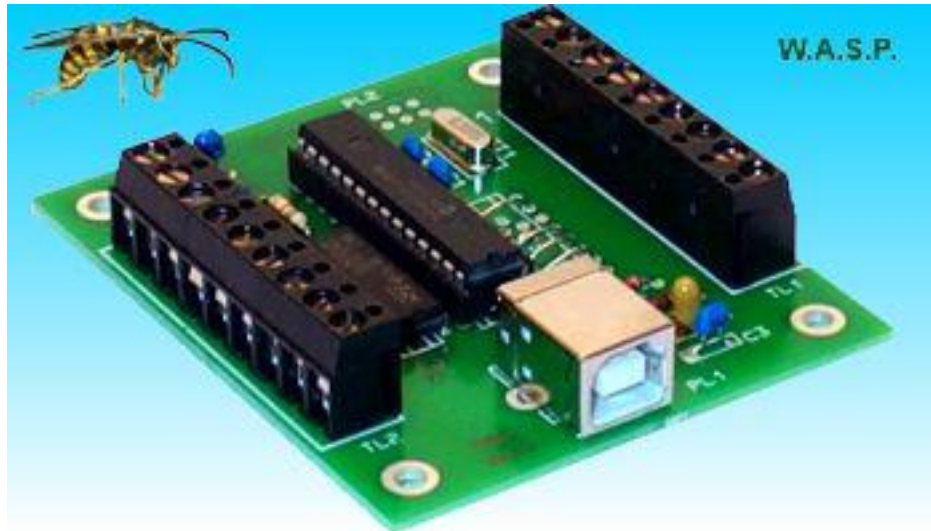
© 2009 Copyright PC Control Ltd.

Contents

1. Introduction
2. Hardware Installation
3. Connecting to WASP
 - a. Analogue Inputs
 - b. Switching Outputs
 - c. Connector Pinouts
4. Software Installation
5. WaspWare Operation
6. Writing your own software for WASP
 - 6.1 Writing your own software in Visual Basic
 - 6.2 Writing your own software in Visual C++
7. Minimum PC System Requirements

1 Introduction

The W.A.S.P. (Windows Analogue Signal Processor) offers a convenient way of linking your PC compatible computer to the real world of analogue and digital signals. It has four analogue inputs and 7 digital outputs. The analogue inputs accept voltages in the range 0-5v and the digital outputs are capable of switching on and off a wide range of devices directly (including high voltage loads up to 50v DC). It is connected to the PC via a standard USB interface requiring little or no installation expertise. i.e. the PC will recognise it as soon as attached and automatically configure itself to make use of the WASP.



The WASP is designed to enable a control system to be implemented where the digital outputs can be turned on and off in response to variations in the analogue inputs. It is supplied with WaspWare application software which in most cases will be all that is needed to implement such a control system. However, a DLL (dynamic link library) is provided to allow programmers to write their own applications and easily access its facilities without having to know the details of USB operation etc.. Although the four analogue inputs are identical they can accommodate a wide range of sensors to suit a wide range of possible applications. These would include the measurement of temperature, pressure, light, sound, position, velocity, acceleration, humidity etc, etc.... The digital outputs can directly connect to an equally wide range of devices including motors, relays, lights, alarm sounders, etc. and with the inclusion of relays the overall power rating of what can be switched becomes limitless.

2 Hardware Installation

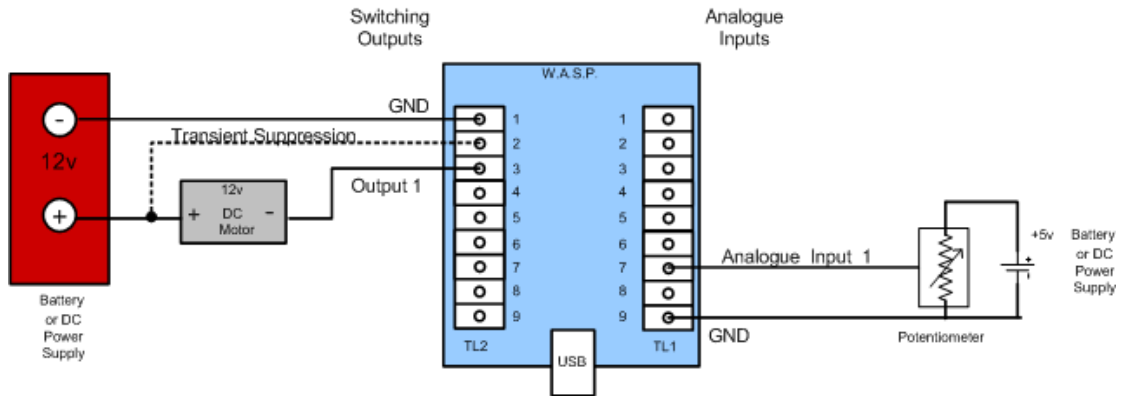
Simply connect the WASP to any available USB port (*This will require a standard USB cable*). Although it will operate from bus powered hubs it is recommended that you connect it to a primary USB socket or a self powered hub. This allows WASP to take full advantage of the available 500mA from such a connection (bus powered hubs are limited to 100mA). Windows operating system will automatically detect and install the appropriate device drivers. The WASP is regarded by Windows as a standard HID (Human Interface Device) which makes it very easy to install.

3 Connecting to WASP

Connecting external sensors and devices to WASP is made simple by the provision of screw terminals. This avoids the need for soldering.

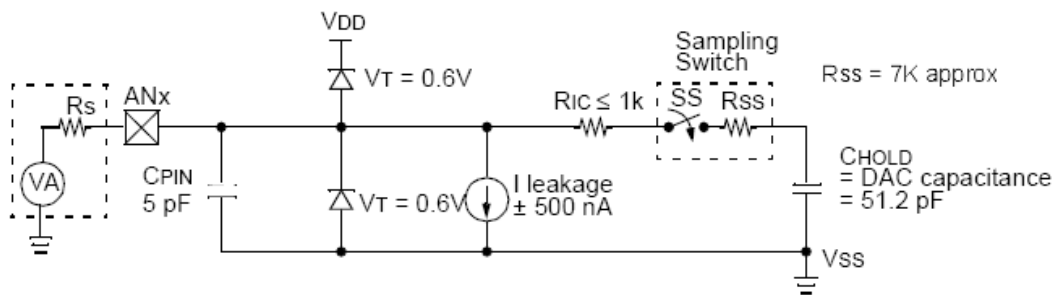
3a Analogue Input Signals

The analogue inputs require a voltage source in the range 0v to +5v relative to the 0v terminal on TL1 (TL1-9). This range can be changed to suit the sensor connected (see Sensitivity description later). A very simple test connection can be made as shown below. This uses a potentiometer as a voltage divider to generate the varying voltage which is applied to analogue input 1 (TL1-7).



A simple example of 1 analogue input and 1 switching output connected to WASP

For those who require more detail about the electrical characteristics of the analogue inputs an electrical model is shown below. As a rough guide the most important thing is to ensure that the source impedance (R_s) of the analogue signal (VA) is less than 10k.



Legend	CPIN	= input capacitance
	VT	= threshold voltage
	I leakage	= leakage current at the pin due to various junctions
	Ric	= interconnect resistance
	SS	= sampling switch
	CHOLD	= sample/hold capacitance (from DAC)

3b High voltage switching outputs

Each switching output takes the form of an “open collector” driver. For these outputs to operate correctly it is necessary to link the 0v (or Ground) connection of the WASP (screw terminal TL2-1) to the 0v connection of the external power supply which is being used to “drive” the device to be controlled (eg a motor, solenoid, lamp etc...).

The device being controlled is then connected between one of the WASP’s control outputs (e.g. Output 1 on screw terminal TL2-9) and the external positive end of the supply (e.g. +12v).

When the WASP output is turned on (by WaspWare or by your own software) the terminal becomes a low impedance path to ground and current flows in the external circuit through the attached device. For low power non-inductive devices such as lamps, that is all that is necessary. If you are directly attaching an inductive load (such as a motor or relay) it is advisable to take precautions against switching transients. Switching transients are spikes in the voltage that occur when an inductive load is turned off and can be high enough to cause damage to attached circuitry. WASP has built in facilities to suppress these transients by using a suppressor diode connected to screw terminal TL2-2. To make use of this simply connect terminal TL2-2 to the external positive supply being used to “drive” your inductive load.

When using voltage suppression as described it is essential that only one external supply voltage is used. i.e. without suppression there is no reason why each of the different control outputs could not be driven by different supplies so long as their 0v connections are common. However, with transient suppression, they must all operate on the same external supply.

The output switching components used by WASP are DS2003 High Current / High Voltage Darlington Drivers. These devices are capable of being switched up to 50v and 350mA. However, although capable of these operating limits, the recommended application of the WASP is for voltages up to 24v. The current capability on each output is 350mA. As the DS2003 data sheet suggests, this can be extended by connecting outputs in parallel. If doing this, it is obviously essential that your control system ensures that all outputs that are paralleled are always in the same state (on/off) at the same time or you run the risk of one output taking all of the current and exceeding maximum limits. You must also take account of the overall power handling capability of the DS2003 when using multiple outputs each with high current. Refer to the graph of “peak collector current vs duty cycle and number of outputs” in the DS2003 data sheet included on the installation CD.

.....

3c Connector Pinouts

Pinout of the analogue inputs on screw terminals (TL1)

Pin	Signal description
1	NC
2	Analogue input 4
3	NC
4	NC
5	Analogue input 3
6	Analogue input 2
7	Analogue input 1
8	NC
9	GND

Pinout of the High Voltage Switching Outputs On Screw Terminals(TL2)

Pin	Signal description
1	GND
2	Transient Suppression
3	Switching Output 1
4	Switching Output 2
5	Switching Output 3
6	Switching Output 4
7	Switching Output 5
8	Switching Output 6
9	Switching Output 7

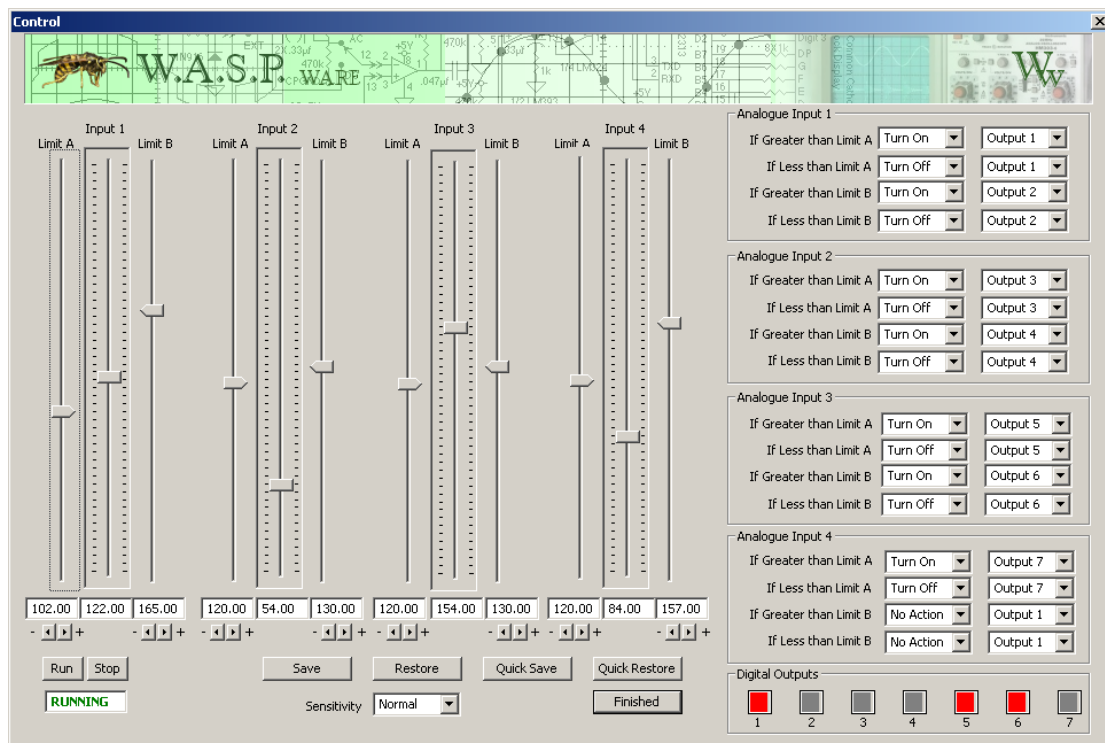
For both terminals, terminal 9 is the one nearest to the terminal number label (i.e. TL1 or TL2)

Software Installation

To install “WaspWare” simply insert the installation CD into your drive and follow the on-screen instructions. If the installer does not start automatically then open Windows Explorer, browse to the CD drive and double click on the “setup.exe” program.

5 WaspWare Operation

When the WaspWare application is started, click on the “Run Wasp” menu option to initiate the control dialog box (shown below). The first time WaspWare is run the customary disclaimer agreement will ask for your agreement. Click “I Agree” to never see it again and to start WaspWare.



When the control box is displayed it will initially be in the “stopped” condition. i.e. not scanning the analogue inputs or controlling the switching of the digital outputs. This can be changed by pressing the “Run” button. The current state of the control system is always displayed in the status message box i.e. “Stopped” or “Running”.

When running, the four analogue inputs will be read and displayed on the four vertical sliders. The height of the sliders will correspond to the amplitude of the input analogue signals. There will also be a numeric readout of this amplitude at the bottom of each slider. The numeric readout is the exact measurement made by the “Analogue to Digital Converters” on the WASP and is in the range 0 – 255 (i.e. 8 bits).

Associated with each of the analogue sliders are two limits. These are shown as slider pointers on each side of the analogue input and labelled “Limit A” and “Limit B”. These can be positioned anywhere over the full range of the analogue input by clicking and dragging with the mouse pointer or by clicking on the +/- keys at the bottom of the limit. A numerical readout of the limit position is also shown

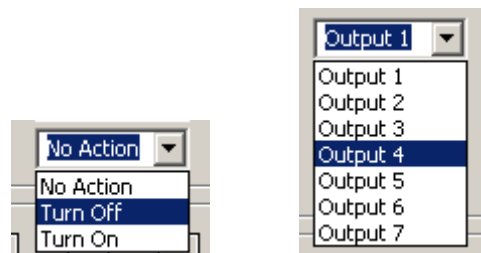
above the +/- keys. These limits form thresholds to compare the analogue inputs against with the intention of switching an output on or off when a limit is crossed.

Although the analogue inputs are designed to operate over the range 0-5v, this can sometimes be inconvenient when a sensor only generates a signal in the range 0-2 volts (for example). Although the analogue reading will be correct it will not be using its full range and therefore resolution to display this. To change the range of the A/D converter simply choose "High" from the sensitivity box. This will change the range to 0-2v. For highly specific applications, other ranges can be accommodated by a component change on the board. Specifically, changing Zener Diode D1 for one with a different reference voltage will change the digitiser range to that reference voltage whenever the sensitivity is set to high. Obviously this has to be less than 5v

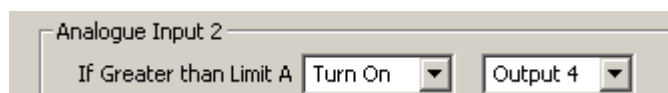
On the right hand side of the control display area are the switching output controls. They are divided into four areas corresponding to each of the four analogue inputs. They offer a very flexible way of specifying what to do with an output when the analogue input crosses one of its limits. Considering analogue input 1 and taking the first option as an example, the control option can be specified as

If greater than limit A [TURN ON] [OUTPUT 3]

This will do exactly what it says, i.e. it will turn on switching output 3 whenever analogue input 1 is greater than the specified limit A.



By choosing the appropriate term from the drop down boxes, any of the 7 outputs can be switched on or off by any analogue signal going above or below any limit.



Further more there is no restriction on an output being associated with any particular signal, i.e. there is nothing to prevent you turning on an output when one analogue signal crosses a limit and off when another analogue signal crosses a limit. If a conflict exists producing the situation where, for example, one output is turned on by one analogue signal and off by another at the same time, it is resolved by processing the decisions top to bottom. In this way whatever decision is made, for an output, it will be based on the last condition which sets it in top to bottom screen order. By careful choice of these options, a very wide range of possible control systems can be created.



To assist in developing the control system, the on/off status of the switching outputs is displayed on the bottom right hand side of the display area.

Once you have set the position of the limits and chosen the controlled output options, you can save the entire configuration to disk by clicking on the save button and specifying a name for the file. Conversely previously saved configurations can be restored by clicking the "Restore" button and choosing the desired file. As a convenience "Quick Save" and "Quick Restore" buttons are provided which do the same as the save and restore but to a fixed filename "waspcurrent.wsp".

The control system can be terminated by clicking on the "Finished" button and closing the main application window.

6 Writing your own software for WASP

Although WASP comes with its own software (WaspWare) to allow the beginner to start using it in home automation projects very quickly, it also comes with a DLL interface to allow the intermediate and advanced user to write their own programs for it. The DLL provides a general purpose interface that greatly simplifies the task of writing programs for a USB device. Without the DLL it can be tricky manipulating the USB comms into sending and receiving messages to and from a device which can easily be plugged and unplugged at any time. Although the DLL was written in 'C' it can be used (called) by programs written in a number of popular languages including BASIC (visual BASIC etc..). Presented below are descriptions of how to use the DLL with Visual Basic and Visual C++.

6.1 Programming W.A.S.P. in Visual Basic

There are four library functions within the DLL.....

[InitWasp\(\)](#) , [SetOutputs \(outputs\)](#), [ReadInputs\(analogue\)](#) and [SetSensitivity\(high\)](#)

[InitWasp\(\)](#) is called somewhere near the start of your program and takes care of all of the USB comms initialisation and prepares the WASP for sending and receiving messages.

[SetOutputs\(outputs\)](#) can then be called at any time during your program to set the output pattern of on's and off's. The parameter Outputs is simply a 32 bit integer value. In Outputs, bit0 corresponds to output 1, bit 1 to output2, etc... In each case a logic value of 1 turns the output on and a value of 0 turns it off. For example the statement below would turn on the first three outputs...

[SetOutputs \(7\)](#)

and the following would turn on outputs 1, 2, and 4

[SetOutputs \(11\)](#)

[ReadInputs\(analogue\)](#) can also be called at any time during your program to read the current values of the analogue inputs. When you call this function the WASP performs an analogue to digital conversion on the 4 inputs and stores the results in 4 (32 bit) integers. The integers are actually the elements of the array called analogue. This array has been passed By Reference to the wasp dll by your program, allowing the dll to modify its contents directly. Your array should be declared as follows...

[Dim analogue\(4\) As Integer](#)

The values stored in these array elements will correspond to the analogue inputs representing 0 to 5volts as 0 to 255. For example if the analogue inputs are 1v, 2.5v, 4v and 5v respectively the contents of the array analogue after the

[ReadInputs\(analogue\(0\) \)](#)

function call will be....

analogue(0) = 51, analogue(1)=128, analogue(2)=204 and analogue(3)=255

The only other thing that a VB program must do is to declare the functions that it is going to use within the DLL and the name of the DLL itself. This must be done at the start of your program or at least before any references to the two functions are made. The following is a program excerpt showing how this is done...

```
Declare Function InitWasp Lib "wsp.dll" () As Boolean
Declare Function SetOutputs Lib "wsp.dll" (ByVal outputs As Integer) As Integer
Declare Function ReadInputs Lib "wsp.dll" (ByRef analogue As Integer) As Integer
Declare Function SetSensitivity Lib "wsp.dll" (ByVal high As Integer) As Boolean
```

The first declaration states that the function `InitWasp` has no parameters, is found in `wsp.dll` and returns a boolean value. The second states that `SetOutputs` has one integer parameter passed by value rather than reference, is found in `wsp.dll` and also returns an integer value. The third declaration states that `ReadInputs` has one parameter which is a reference to an integer and returns an integer value. The reference to the integer is actually the reference to the first integer in an integer array, with the subsequent 3 array values also being used to store analogue results. The fourth states that `SetSensitivity` has one integer parameter passed by value rather than reference, is found in `wsp.dll` and returns a boolean value. It should be noted that the

```
Lib "wsp.dll"
```

lets the program know where to find the `wsp.dll` file. When written like this it assumes, since there is no path information, that the `wsp.dll` file can be found in the windows system directory `c:\windows\system32`. If you like you can copy the file `wsp.dll` on the installation disk to the `system32` directory and the above statement will work perfectly. Alternatively you can copy the file to some other location and give that location in the declaration as in the example below...

```
Declare Function InitWasp Lib "c:\mylibrary\wsp.dll" () As Boolean
```

To speed up your development of software for the WASP a complete working example is provided in the directory `VBwasp`. It creates a very simple form based program that has individual buttons for various functions such as initialising the WASP, reading the analogue inputs and setting various patterns on the outputs. This has been written using Microsoft Visual Studio .net and the directory contains the full workspace (solution) details to allow you to immediately open and start editing or running this application.

Even if you don't have Visual Studio, the source code is virtually self explanatory with the most relevant sections being in `"Form1.vb"` which can even be opened in a simple text editor such as notepad.

6.2 Programming the WASP in Visual C++

Ignoring some of the formalities in the construction of a Visual C++ program for the windows environment the techniques in using “wsp.dll” consists of four main tasks....

6.2a Loading the DLL into memory

Before any functions within the DLL can be used it is necessary to instruct windows to load it into memory. This is done by calling the LoadLibrary() function. i.e.

```
.....  
HINSTANCE WspHandle;           // declaration of variable to hold the handle to the dll  
....  
WspHandle = LoadLibrary(“wsp.dll”); // load the dll into memory and return handle
```

The declaration of the variable WspHandle used to store the handle to a DLL , uses a built in type definition which is called HINSTANCE in this particular ‘C’ compiler, but you should use the appropriate one defined in your own compiler for this purpose.

The LoadLibrary() function returns a handle to the DLL if the load is successful otherwise NULL. Ideally your own program should check for a NULL returned and give an error message. Make sure the function parameter is the full pathlist to where you copied the bee.dll file from the installation CD.

6.2b Get the addresses of the functions within the DLL

Using the DLL handle returned above you can now obtain pointers to the functions within the DLL. Using the following

```
TypeInitWasp      InitWasp;  
TypeSetOutputs    SetOutputs;  
TypeReadInputs    ReadInputs;  
TypeSetSensitivity SetSensitivity;  
  
.....  
InitWasp = (TypeInitWasp)GetProcAddress( WspHandle, "InitWasp");  
SetOutputs=(TypeSetOutputs)GetProcAddress(WspHandle, "SetOutputs");  
ReadInputs=(TypeReadInputs)GetProcAddress(WspHandle, "ReadInputs");  
SetSensitivity=(TypeSetSensitivity)GetProcAddress(WspHandle, "SetSensitivity");  
  
.....
```

The TypeInitWasp, TypeSetOutputs, TypeReadInputs and TypeSetSensitivity type definitions are contained in the header file “wd.h” and defines the correct type of function pointer to reference the DLL function.

Wsp.h should be included in your own source file eg.

```
.....  
#include “Wsp.h”  
.....
```

The call to GetProcAddress() returns a pointer to this function if found within the DLL otherwise NULL. Once the functions pointers have been obtained in this

way the internal functions within the DLL are simply accessed like ordinary function calls e.g.

```
.....  
InitWasp();  
SetOutputs(0x12);  
ReadInputs(Inputs);  
SetSensitivity(1);  
.....
```

6.2c Initialising The DLL

Once the addresses of the DLL functions are obtained as above the remaining functions required to use them are very simple. The first step is to initialise the DLL using....

```
int status;  
.....  
status = InitWasp();
```

Your program should check to see if a value of zero has been returned by InitWasp(). Any other value indicates an error. e.g. WASP not connected etc...

6.2d Using the ReadInputs() Function

The ReadInputs() function initiates an Analogue to Digital conversion function within the WASP and returns the values converted to the array supplied as a pointer parameter. For example if the four analogue inputs have voltages of 1,2,3 and 4 volts respectively then the following call.....

```
.  
.   
int Measurements[4];           // declare the array to hold the results  
.   
.   
ReadInputs(Measurements);     // call ReadInputs function passing the  
                               // address of the results array
```

..... will result in the following values being present in the array

```
Measurements[0] = 51  
Measurements[1]=102  
Measurements[2]=153  
Measurements[3]=204
```

6.2e Using the SetOutputs() Function

The SetOutputs() function simply applies the pattern of 1's and 0's in the supplied parameter directly to the outputs. For example: to create a pattern of alternate on and off over all outputs use..

```
SetOutputs(0x55);           // hexadecimal number
```

Or to turn on just output 1 only use ...
`SetOutputs(0x01);` // hexadecimal number
Etc.....

More generally.....

```
int bits;  
.....  
bits = 0x12; // hexadecimal number  
SetOutputs(bits);  
.....
```

This example will turn on outputs 2 and 5.

6.2f Using the SetSensitivity() Function

The SetSensitivity() function allows the voltage range for the analogue to digital conversion to be changed. Normal sensitivity is for an 8 bit conversion over the range 0 to +5volts. In other words 0 volts will be converted to a value of 0 and +5volts to a value of 255. Setting this to normal requires calling the function with a parameter of '0' i.e.

```
SetSensitivity(0);
```

To use a higher sensitivity call the function with a parameter of 1 i.e....

```
SetSensitivity(1);
```

The range of conversion is now set by a zener diode on the WASP board which provides a new reference voltage. As supplied this is 2.4volts. Therefore a value of 255 will be produced for 2.4volts on the input. This scale change applies to all inputs. If another range is needed the zener diode can be changed for one suitable for your own purposes (obviously not exceeding 5v).

Although this only gives a glimpse of the possibilities of writing your own programs, it should be apparent that the use of the DLL functions greatly simplifies this process. It frees the programmer from the task of getting to know the fine details of programming USB interface communications and lets him concentrate on the main function of reading and converting analogue inputs and controlling switching outputs.

7 Minimum PC System Requirements

WASP and WaspWare software do not require a high spec PC for correct operation, but the following system is suggested as a sensible minimum

Processor	500MHz Pentium
Memory	64MB
HDD	10MB free space required
Operating System	Windows 2000, XP or Vista
Screen Resolution	1024x768 (256 colours)
Interface	One free USB socket (1.0 or 2.0)

WARNING: The WASP adaptor board is intended for DC voltages less than 50v. It should not be connected directly to mains voltages under any circumstances

Terms of Use for all Goods Supplied

Definitions

'Supplier' shall mean PC Control Ltd.

'Buyer' shall mean the person, company or any other body that purchases or agrees to purchase Goods.

'Goods' shall mean all goods and services which the Buyer agrees to buy from the Supplier including replacements for defective Goods, hardware, documentation and software products licensed for use by the Buyer.

Use of the Goods in any way by the Buyer constitutes acceptance of these terms and conditions.

Terms and Conditions

1. The Goods are intended to be part of the buyer's own design of apparatus and not a finished product in their own right.
2. The Goods supplied are not to be used in any design where there is a risk, however small, either directly or indirectly, of death or personal injury.
3. The Buyer will be responsible for ensuring the fitness for purpose of the Goods for the Buyer's application.
4. To the extent permitted by law, the Supplier accepts no liability whatsoever or howsoever arising in respect of loss, damage or expense arising from errors in information or advice provided whether or not due to the Supplier's negligence or that of its employees, agents or sub-contractors save for any loss or damage arising from death or personal injury.
5. To the extent permitted by law, the Supplier shall not be liable to the Buyer by reason of any representation (unless fraudulent), or any implied warranty, condition or other term, or any duty at common law, or under the express terms of any Contract with the Buyer, for any indirect, special or unforeseen loss or damage (whether for loss of profit or otherwise), costs, expenses or other claims for compensation whatsoever (whether caused by the negligence of the Supplier, its employees or agents or otherwise) which arise out of or in connection with the supply of the Goods or their use or resale by the Buyer.
6. The entire liability of the Supplier under or in connection with the Contract with the Buyer shall not exceed the price of the Goods except as expressly provided in these terms and conditions.
7. These terms are an important part of the full terms and conditions of business as published on the website at www.pc-control.co.uk/general-terms.htm which also apply.

If you cannot agree to the terms and conditions of use of the WASP then you should return the WASP to the supplier within 7 days of receipt to receive a refund. Your use of the board or the associated software in any way whatsoever will be regarded as an acceptance of these terms and conditions.