
VPS_P18 Getting Started

---BitCraft-----

Preface

This document is intended to be a quick introduction to the basics of VPS_P18 rather than a lengthy description of the different options available. The introduction will be in the form of describing the making and testing of a very small application for the PIC® microcontroller. The 'application software' that we are going to write, or rather 'draw', is shown in **Figure 1**, where a LED, tied to PORTB pin 0, will flash or stop with every alternate operation of a push-button connected to PORTB pin 4. The circuit diagram required for the application is shown in **Figure 2** but we will in this document take the easy way out by using the simulator included in VPS_P18

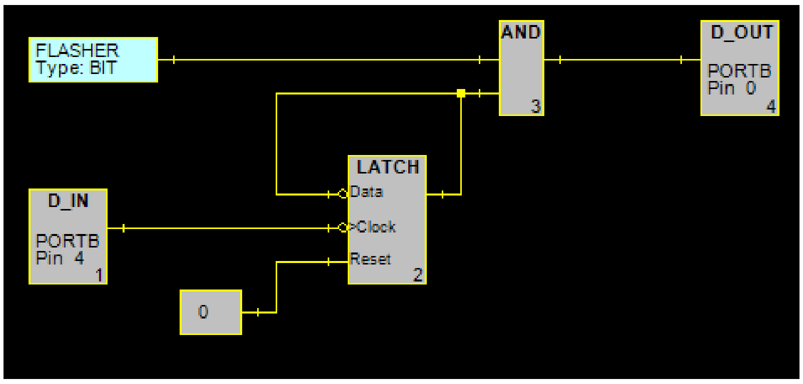


Figure 1. Application Software

The document also contains ***Notes*** with additional information or explanation of related topics.

1.0 Contents

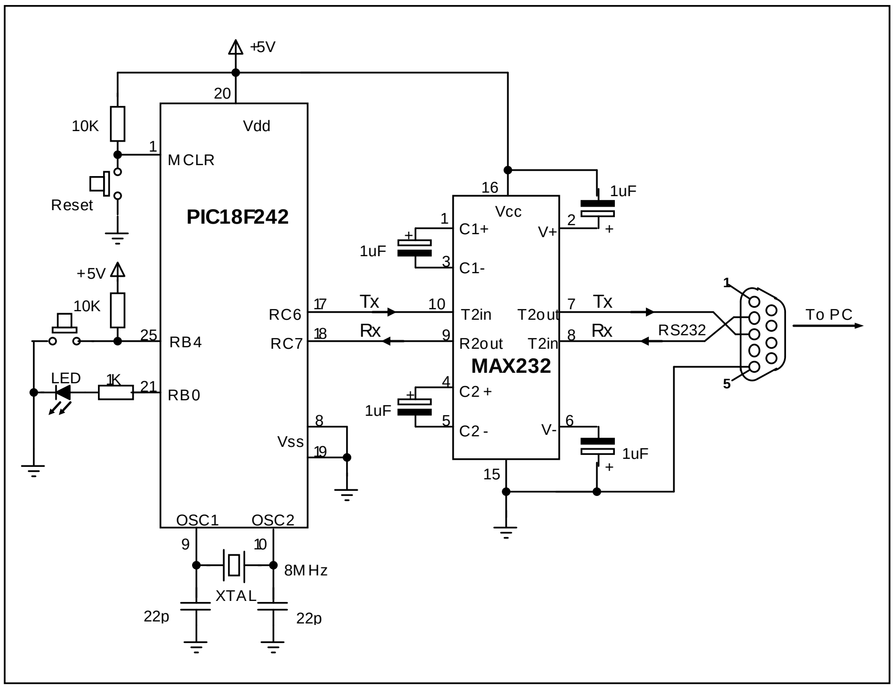


Figure 2. Application circuit

2.0 What is VPS_P18

VPS_P18 is a Windows®-based development application for specific members (See **10.0 Hardware for the project**) of the PIC18F range of microcontrollers from Microchip™ Technology Incorporated. A graphical approach is used where the application/software is drawn in diagram form using the built-in function block elements which you “wire” together. No knowledge of a programming language is required and the package will compile the function block diagrams application into a ready-to-burn .hex file.

The simulator will emulate the execution of your function block application, although the simulators performance is PC dependent. On an i7 performance is about 1.5 times slower than real-time for a one code page application.

The ‘Lite’ version of VPS_P18 is limited to 2 code pages - about 40 function blocks. Should you want to use more code pages you must obtain a registration key from bitcraft@global.co.za. A registration key for a machine is available free of charge if you use it for non-commercial purposes.

3.0 Installing VPS_P18

Requirements:

- Intel® Pentium (or better) class PC running Microsoft Windows® XP, 7 or 8.
- Microsoft’s .NET Framework 4 (available as a download from Microsoft.com.

Installation:

Uninstall any previous versions of VPS by running C:\LLG_CB\unins000.exe.Then run the executable file VPS_P18setup2_05.exe to install the application. The installation will place the following icon on your desktop:

4.0 Starting VPS_P18

Start VPS_P18 using the above icon or by executing C:\LLG_CB\VPS_P18.exe to obtain the window as shown in **Figure 3**.

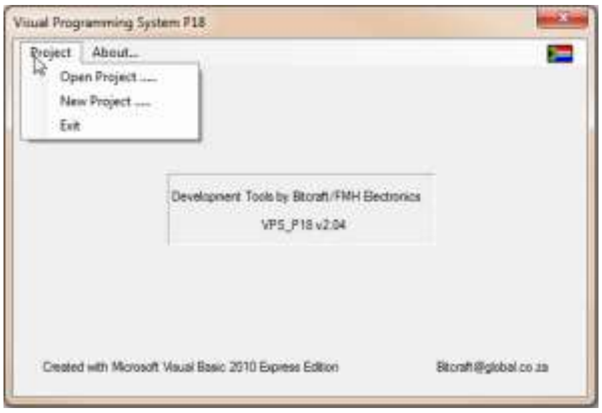

Visual Programming System P18
Project
About...
Open Project .....
New Project .....
Exit
Development Tools by Bitcraft/FMH Electronics
VPS_P18 v2.04
Created with Microsoft Visual Basic 2010 Express Edition
Bitcraft@global.co.za

Figure 3. VPS_P18 Start-up screen

5.0 Creating the Project

Select NewProject and create your project file It is recommended you keep each project in its own directory. A project file has the extension .lcb.
Once you have entered the project file name in the File Dialog pop-up, click the Save button. The file is created and you will now be presented with VPS_P18’s desktop

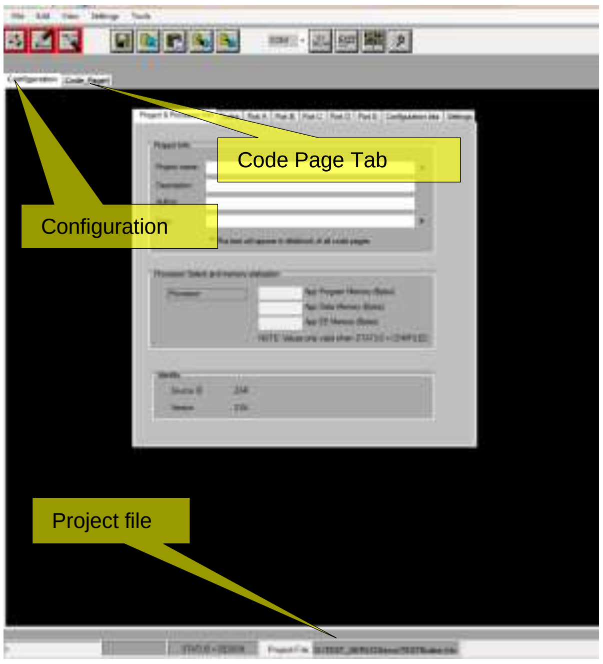


Figure 4. Config. Desctop

(**Figure 4**) with the Configuration tab selected by default. This tab contains a secondary tabcontrol object with its **Project & Processor Info** tab selected. This tab page is provided for your project information Use these fields so you can later identify your project. It also shows the memory usage of your project, which now should be zero because you have not entered any program, or rather function blocks. The tab to the right of the Configuration tab is named **Code_Page1**. A project will contain at least one code page. Code pages are the work surfaces on which you place the function blocks of your application.
At the very bottom of the screen is a status line showing, amongst other information, the path and name of your project file.

6.0 Creating the Function Block Application

6.1 Code Pages

The function block diagram(s) of a project is contained on one or more code pages. There is no real limit to the number of code pages you can use. Each code page is in the form of a tab page; click on the tab and you are presented with the function block contents of that page. The name on the tab of a code page can be changed. For our project we will use two code pages, and we will leave the tab names at their default text.

6.2 The Task Info function block

Each code page contains by default a ‘task info’ Block as per **Figure 5**, located in the bottom left corner. This block’s purpose is so you can select which task the code on the page must be assigned to. Another useful function of this block is that it displays the number of

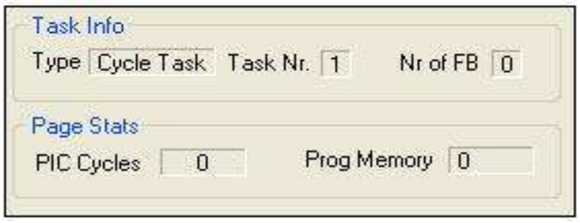

Task Info
Type: Cycle Task   Task Nr.: 1   Nr of FB: 0
Page Stats
PIC Cycles: 0   Prog Memory: 0

Figure 5. Task Info Function Block

cycles the microcontroller is going to use to execute all the function block code on the page and also the program memory (in bytes) that it will occupy.

NOTE:
Two types of tasks are provided, Cyclic and Time tasks. Cycle tasks are executed in a round-robin strategy while time tasks are executed at user selectable intervals of 10, 20, 40, 50, 100, and 200mSec. The system provides for 8 cycle, and 7 time tasks.
NOTE:
You can assign more than one code page to a particular task, and the code pages need not be in a particular order as far as task numbering goes.

When you now select the tab Code_Page1 you will get an empty page with only the Task Info function block in the bottom left corner of the screen and a title block on the right. Double click on an empty spot in the Task Info function block frame to obtain the popup dialog for editing of the task settings for the current code page.

NOTE:
Depending on your screen resolution you might have to use the scroll bars at the bottom and right sides of your screen to properly view the Task Info function block. The recommended display resolution for VPS_P18 is 1280 x 1024.

For the purpose of this exercise make the task type Cycle Task and select Task Nr equal 1. **Figure 5** is a screen shot of how the task info function block should appear after the above settings have been implemented.

6.3 Add function blocks to a code page

Figure 6 is a screenshot showing how the drop-down menu, obtained by a right mouse click, is used to select Add Function Block>Logic>D_LATCH FB17. A click on the item will place the function block on the code page attached to the cursor. Move the cursor towards the middle of the page and place the FB with a left click.

NOTE:
Should you want to delete any item (Function Block or line) simply select the item and use the Delete key..

The D_LATCH (FB17) Function Block will transfer and latch the logic state of its Data input pin to the output pin whenever a leading edge is detected on its Clock input pin, unless of course the Reset pin is at logic zero, in which case the output is held zero.

NOTE*:*
There is documentation available that describes the function of each of the function blocks in detail, except for those which the function/operation is pretty obvious.

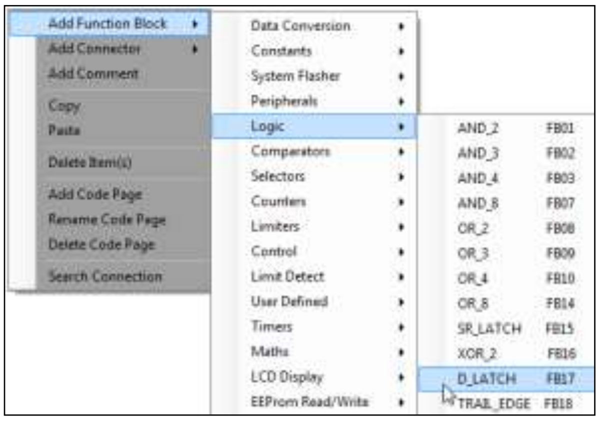


Figure 6. Right Click Menu

Add the following function blocks to the code page:
1. Digital Input function block (Add Function Block>Peripherals>D_IN FB29).
2. Digital Output function block (Add Function Block>Peripherals>D_OUT FB28).
3. 2-Input AND Gate function block (Add Function Block>Logic>AND_2 FB01)

Add the following off-page connector to the code page:
1. Connector function block (AddConnector>Read Connector FB200).

Add the following constant value to the code page:
1. Boolean constant (Add Function Block>Constants>Boolean FB210)

The Task Info function block now indicates the processor cycles and memory requirements for this code page.

NOTE:
As indicated on the circuit diagram an 8MHz crystal is used and together with the PLL the PIC® is clocked at a frequency of 32MHz. Each processor cycle uses 4 clock cycles, so one processor cycle time is 0.125 micro second

Select and drag each function block so the layout looks something like **Figure 7**.

NOTE:
Multiple items can be selected (when not in Insert Line or Edit Line mode) by dragging a rectangle around them. Holding down the Shift key to deselect any of the selected items, or select additional item with a mouse click.

Notice how all function blocks, except the light blue read connector and the Boolean constant, contain a number in the bottom right corner. This number

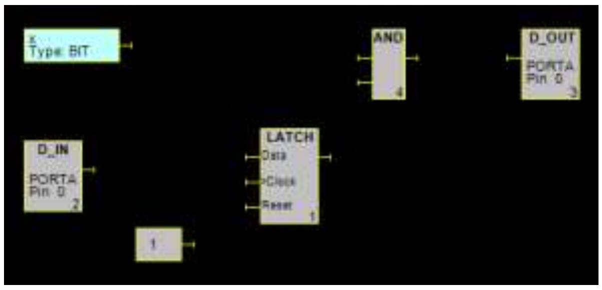


Figure 7. Placing Function Blocks

indicates the sequence in which the final code will be executed in the simulator and the PIC® microcontroller.

NOTE:
The execution sequence of function blocks is important. If the blocks are not executed in the direction of the signal flow (left to right) it will take more than one execution of a task for the signal to propagate from left to right, and will therefore influence the response time of your application. Function blocks without execution sequence numbers do not generate any executable code.

A double click on a function block will open a dialog pop-up where you can change its execution sequence number and other parameters particular to the selected function block. Like in the case of the D_IN and D_OUT blocks where we must change the Port and pin setting from the default to what is required by our application's functional specification. Do not forget to change FB210, the Boolean constant. In our application it will be connect to the Reset input of the LATCH, and as we do not want to reset this latch all the time you must change FB210's default value by a double click on the block to obtain the dialog pop-up and then modify the setting to **0 (FALSE).**

NOTE:
When you change a function block's execution sequence number the system will attempt to automatically renumber the other blocks on the page. This saves you the effort of renumbering all downstream FB's when you insert a new FB amongst existing ones.

Double click on the read connector to obtain a dialog box and enter the Signal Name as FLASHER. Make sure the signal data type is selected as BIT. This read connector is going to have a counterpart, a write connector located on a different code page, the code of which we will get to shortly.

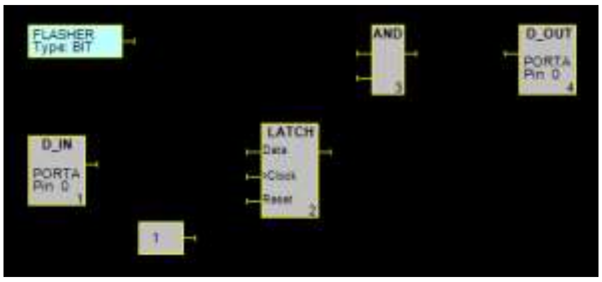


Figure 8. Execution Sequence Nrs.

NOTE:
Read and write connectors are used to interconnect signals from one code page to another. Read connectors also serve another purpose in that when on-line (or Simulator) you can change the signal value, but only if there is no Write connector on the current or any another code page, with the same name, because if there is then the value supplied by you will be overwritten by the application code. Choose your signal names carefully, assembly errors will be generated if you use assembler reserved names. .

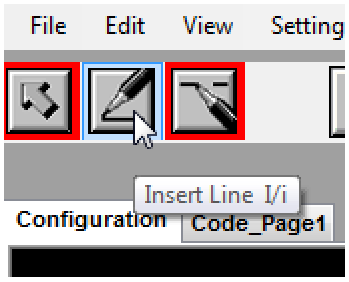


Figure 9. Insert, Edit line modes

After the necessary changes your function block application should look like **Figure 8**, ready for a bit of virtual soldering of the interconnecting wires!

6.4 Connecting Function Blocks

Connecting lines are inserted and edited by selecting the **Insert Line** and **Edit Line** modes using the icons in the tools bar at the top left of the screen as indicated by the **Figure 9** screen shot where the Select, Insert Line, and Edit Line selection buttons are shown. The cursor will change according to the selected mode – an arrow for insert and up-arrow for edit.

Select the Insert Line mode and draw the lines to connect the function blocks using mouse-down and drag. A line will start at the mouse-down position and will end at the mouse-up position. Markers where lines join are inserted automatically, but notice that only 'T' junctions are allowed and only horizontal and vertical lines are possible.

Your project **Code_Page 1** should look something like **Figure 1** once you have done all the interconnections. Notice the 'inversion' circles on the Data and Clock inputs of the LATCH function block. All Boolean inputs of function blocks can be inverted by holding down the Alt button and then click on the input pin of the function block near the rectangle body. Do the same thing to remove the inversion.

NOTE:

While on the subject of connecting function blocks... Each input and output signal of a function block has a data type. VPS_P18 makes use of BIT, BYTE, UINT (Unsigned Integer), INT (Signed Integer), and FP32 (Floating Point) data types. When you connect input/output pins of different data types together the compiler will produce an error message. The data types of the input/output pins of a function block can be identified by holding down the Shift key and then move the mouse cursor over the function block.

Now it is time to add the function blocks that will generate the pulsing signal. Two methods are available to obtain a pulsing signal. The first method is simply to use one of the internal flasher signals generated by the operating system, but that will be far too easy! We will use the second method which requires the PULSER function block (FB21). If you try to add a FB21 to the current code page (Add Function Block>Logic>PULSER FB21) you will get an error message saying that the PULSER FB can only be used in a Time Task. So use the Right-Click-Menu, select **Add Page** and observe that another code page has been added with its tab name **Code_Page2**. Select the tab of this new page to get to it, and modify the Task Info blocks Task Type to Time Task and make the Task Number 1.

Add the following to Code_Page2:

-PULSER function block (Add Function Block>Logic>PULSER FB21).
-Boolean constant function block (Add Function Block>Constants>Boolean FB210).
-Write connector block (Add Connector>Write Connect FB201)

Double click the PULSER function block to obtain its dialog box, which will tell you the block will generate pulses at a rate of 2.5 per second. This is a nice rate so leave the Scan-down factor setting at 1. The Cycle Time indication of 200mSec is simply information showing you the execution rate assigned to Time Task 1.

NOTE:
All 7 the Time Tasks default to a cycle time of 200mSec. When you select the Configuration tab and then the Task tab you are presented with a dialog where you can change the cycle times. Cycle times of 10, 20, 40, 50, 100, and 200 mSec are available.

CAUTION:
If you change the cycle time of a time task then you must (re)check all the time-dependant function blocks executed by this task because most have settings that depends on rate of execution. Like our PULSER above, the Scan-down rate setting will remain at what you have set it to, but if the execution rate of the task is change you will have a different pulse rate.

Double click the light blue Write Connector. The dialog box indicates the Signal Name as 'x', which is the default. The dialog box also contains a text box containing the names of all the signal connectors in the project, which in our case is only that of FLASHER. Double click on FLASHER, which will update the Signal Name and Data type parameters for you. Click on Apply… to accept the settings.

Insert a line to connect the Enable input of the PULSER function block to the Boolean 1 block, and its output pin to the input of the write connector with signal name FLASHER. Your application should look something like

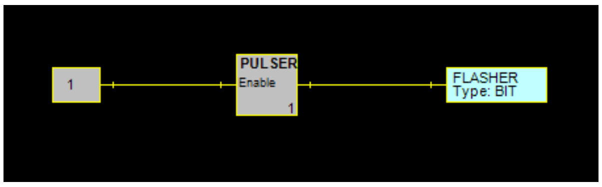


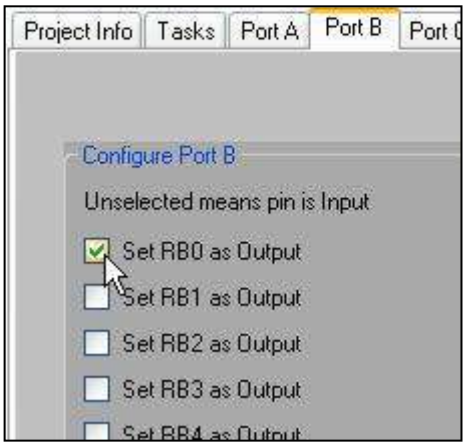
Figure 10.
Figure 10. The Pulse Generator

7.0 Port and Peripheral Configuration

To configure the PIC® microcontroller ports and peripherals you must select the **Configuration** tab and then the required port tab. For our project it is required to only set up PORTB for the LED and push button. Leave the rest of the port settings at their default values.

Figure 11 shows a section of the dialog box obtained after you have selected the Port B tab. It shows how RB0 is selected as Output because it will drive our LED. Remember to click the Apply…. button after you have marked the box to set RB0 as output.

In the **Tools** menu item is a selection named **Task**



Configure Port B
Unselected means pin is Input
☑ Set RB0 as Output
☐ Set RB1 as Output
☐ Set RB2 as Output
☐ Set RB3 as Output
☐ Set RB4 as Output

Figure 11. Port B Setup

Stats. If you click it you will obtain information about the program memory used and execution cycles required for each task of your project.

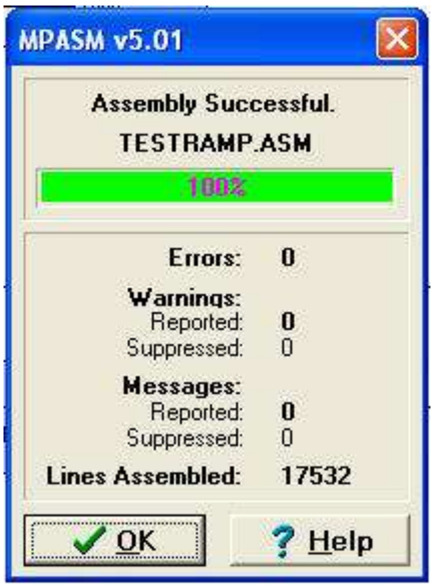
8.0 Compile the Project

Next step is to generate the .hex file for the PIC® controller.

NOTE:
On the status bar at the bottom of the screen there is an indication of the state of the application STATUS = DESIGN. Once you have successfully compiled your project this will change to STATUS =

COMPILED. Your application must be at STATUS = COMPILED to be able to go on-line or use the simulator. The status of an application is saved in the project file.

On the **Tools** menu click the **Compile** item to start the compiling process. After some internal housekeeping it will start Microchip's MPASM Assembler. This is indicated by the Microchip assembler pop-up (**Figure 12**) showing the assembler at work. After a few seconds it should indicates that the assembly was carried out successfully. (No errors or warnings) To continue the compilation process you must click the OK button on the pop-up.


MPASM v5.01

Assembly Successful.
TESTRAMP.ASM
100%

```
            Errors:    0
          Warnings:
           Reported:   0
          Suppressed:  0
          Messages:
           Reported:   0
          Suppressed:  0
   Lines Assembled:    17532
```

Figure 12. Assembler.

The status bar at the bottom of the screen should now indicate STATUS = COMPILED. Another thing you will notice about a COMPILED project is that if you place the cursor over a function block (see **Figure 13**) the register addresses (HEX format) of all input and outputs of that block will be displayed at their respective pins. These addresses you use in on-line mode for the Trend View option discussed in the next section.

The directory where your project file is located should now also contain a **.hex** file. This file you can program into your microcontroller. I use Microchip's MPLAB® with a PICkit™3 programmer.

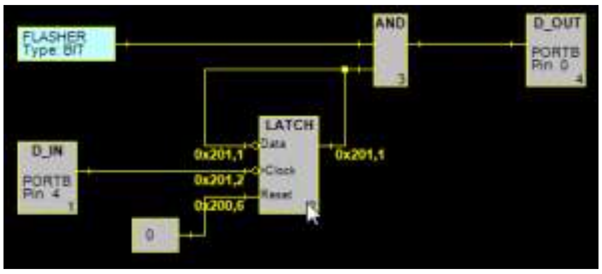


Figure 13. Signal addresses

Next we will use the simulator to verify that the application is functioning as per specification.

9.0 On-Line Monitoring and Debugging of the Application

The simulator (and when using f RS232 communication when the processor is used) allow you to visually monitor the real-time input and output values of function

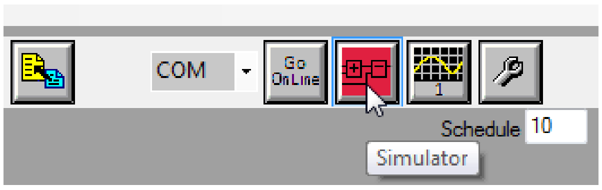


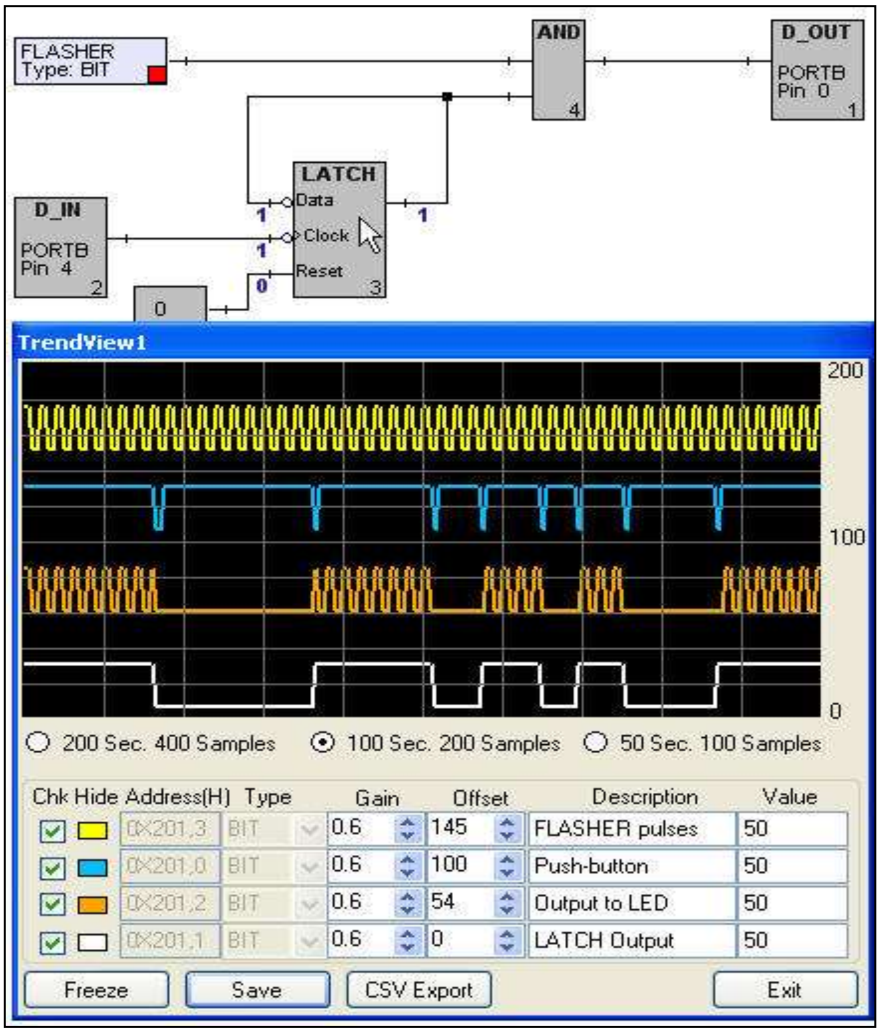
Figure 14. Activate Simulator

blocks, and also supply a 'Trend view' where you can observe the last 400 samples of up to 4 values. (See **Figure 16**).. Development of control strategies is also made easy by the 'register modification' options. The value property of Read Connectors in the application can be modified when in on-line or simulator mode.

Activate the simulator as indicated in in **Figure 14**.
If you now place the cursor over any function block you should see your application in action. Take note that the logic status shown for a Boolean pin that contains an inversion circle is the status before inversion. That means the inversion is done inside the function block.

***NOTE**:*
When in on-line mode with the cursor over a function block VPS_P18 will poll the PIC® for information. If there is no reply from the microcontroller for about 5 seconds an error message is displayed. Make sure you have selected the correct PC COM port number for RS232 communication.

In many a case testing requires observing how variables behave over a period of time and in relation to one another. **Figure 15** demonstrates how this can be done in VPS_P18 using the TrendView facility.



○ 200 Sec. 400 Samples [✓ 100 Sec. 200 Samples] ○ 50 Sec. 100 Samples

Chk	Hide	Address(H)	Type	Gain	Offset	Description	Value
✓	🟨	0X201,3	BIT	0.6	145	FLASHER pulses	50
✓	🟦	0X201,0	BIT	0.6	100	Push-button	50
✓	🟧	0X201,2	BIT	0.6	54	Output to LED	50
✓	⬜	0X201,1	BIT	0.6	0	LATCH Output	50

Figure 15. Testing the application.

10.0 Hardware for the project.

The circuit for our project is straight forward and can be assembled on a piece of Vero board. For the circuit of **Figure 2** you can use a PIC18F242-I/SP, or the PIC18F252-I/SP, both of which are available in 28 Pin DIL packages. For a new project it is recommended to use the PIC18Fxx20-I/SP devices. For applications that require more input/output pins you can use the PIC18F4420-I/SP or PIC18F4520-I/SP processors.. Just make sure your selection of processor on the **Configuration>Project & Processor Info** tab is correct

11.0 List of Function Blocks.

11.1 Logic Functions.

Reference	Description
FB1	AND Gate 2-Input
FB2	AND Gate 3-Input
FB3	AND Gate 4-Input
FB7	AND Gate 8-Input
FB8	OR Gate 2-Input
FB9	OR Gate 3-Input
FB10	OR Gate 4-Input
FB12	Schmitt Trigger
FB14	OR Gate 8-Input
FB15	SR Latch
FB16	XOR Gate 2-Input
FB17	D-LATCH with Reset
FB18	TRAILING EDGE Detect
FB19	LEADING EDGE Detect
FB138	Byte AND
FB139	Byte OR
FB140	Byte XOR
FB141	Byte INVERT

11.2 Timing Functions.

Reference	Description
FB30	ON-Delay Timer
FB31	OFF-Delay Timer
FB32	MONO-Stable Timer
FB33	MONO-Stable Timer retriggerable

11.3 Mathematical Functions.

Reference	Description
FB120	ADD INT Values
FB102	ADD FLOATING POINT Values
FB122	SUBTRACT INT Values
FB103	SUBTRACT FLOATING POINT Values
FB124	MULTIPLY INT Values
FB101	MULTIPLY FLOATING POINT Values
FB126	DIVIDE INT Values
FB100	DIVIDE FLOATING POINT Values
FB105	ABSOLUTE of FLOATING POINT Value
FB129	ABSOLUTE of INT Value
FB170	INTEGRATOR for INT Values
FB171	DIFFERENTIATOR for INT Values

11.4 Comparator Functions.

Reference	Description
FB59	COMPARE for BYTE Values >, =, <
FB92	COMPARE for INT Values >, =, <
FB60	TEST if A>=B for INT Values
FB61	TEST if A=B for INT Values
FB64	TEST if A<B for FLOATING POINT Values
FB65	TEST if A>B for FLOATING POINT Values

11.5 Variable Test Functions.

Reference	Description
FB93	TEST INT Value for +ve, =0, -ve

FB98	TEST FLOATING POINT Value for +ve, =0, -ve

11.6 Counter Functions.

Reference	Description
FB39	8BIT UP-DOWN COUNTER with Limits
FB40	16BIT UP-DOWN COUNTER with Limits

11.7 Selector/Multiplexor Functions.

Reference	Description
FB20	CHANGE-OVER-SWITCH for BOOLEAN values
FB56	CHANGE-OVER-SWITCH for BYTE values
FB72	CHANGE-OVER-SWITCH for INT values
FB78	CHANGE-OVER-SWITCH for FLOATING POINT values
FB41	MUX for 1-of-8 BYTE Literals
FB42	MUX for 1-of-8 INT Literals
FB43	MUX for 1-of-8 UINT Literals
FB44	MUX for 1-of-8 FLOATING POINT Literals
FB79	SELECT 1-of-4 FLOATING POINT Values
FB70	SELECT MAXIMUM of 2 INT Values
FB71	SELECT MINIMUM of 2 INT Values
FB76	SELECT MAXIMUM of 2 FLOATING POINT Values
FB77	SELECT MINIMUM of 2 FLOATING POINT Values
FB107	SELECT 1-of-12 PATTERNS (mainly for 7-segment pattern look-up)
FB108	SELECT 1-of-8 BYTE Literal values
FB109	SELECT 1-of-8 INT Literal values
FB110	SELECT 1-of-8 UINT Literal values
FB111	SELECT 1-of-8 FLOATING POINT Literal values

11.8 Limiter Functions.

Reference	Description
FB83	HIGH LIMITER for INT Values
FB84	LOW LIMITER for INT Values
FB90	HIGH LIMIT DETECT for INT Values
FB91	LOW LIMIT DETECT for INT Values
FB94	HIGH LIMIT DETECT for FLOATING POINT Values
FB95	LOW LIMIT DETECT for FLOATING POINT Values

11.9 Table Look-up (Function Generator) Functions.

Reference	Description
FB168	FUNCTION GENERATOR INT Values (Input 0…1024, Output -32768…32767)
FB169	FUNCTION GENERATOR INT Values (Input 0…32767, Output -32768…32767)

11.10 Communication Functions.

Reference	Description
FB163	I2C WRITE (to Slave, 7Bit address)
FB164	I2C READ (from Slave, 7Bit address)

11.11 Input and Output Functions.

Reference	Description
FB13	1 to 4 CHANNEL HOBBY SERVO DRIVER
FB23	8BIT SERIAL INPUT DRIVER (for 74LS166)
FB24	8BIT SERIAL OUTPUT DRIVER (for 74LS595 or similar)
FB25	ANALOG INPUT
FB26	FREQUENCY COUNTER INPUT (max 32000Hz)
FB27	PWM OUTPUT
FB28	DIGITAL OUTPUT

FB29	DIGITAL INPUT
FB142	DISPLAY BYTE Variable on LCD
FB143	DISPLAY INT Variable on LCD
FB145	DISPLAY FLOATING POINT Variable on LCD
FB146	DISPLAY STRING on LCD
FB147	DISPLAY TIME (HH:MM:SS) on LCD
FB148	DISPLAY TIME (HH:MM) on LCD

11.12 Pulse Generator Functions.

Reference	Description
FB21	LOW FREQUENCY PULSE GENERATOR
FB36	PULSE WIDTH MODULATOR
FB211	FLASHER BITS (System Resource)

11.13 Data Pack/Unpack Functions.

Reference	Description
FB112	PACK 8 Bits into BYTE
FB113	PACK 2 BYTES into INT
FB114	PACK 2 BYTES into UINT
FB115	PACK 4 BYTES into FLOATING POINT
FB116	UNPACK BYTE into 8 Bits
FB117	UNPACK INT into 2 BYTES
FB118	UNPACK UINT into 2 BYTES
FB119	UNPACK FLOATING POINT into 4 BYTES

11.14 Read & Write Data EEPROM Functions.

Reference	Description
FB45	READ BYTE Value from EEPROM
FB47	READ INT Value from EEPROM
FB49	READ UINT Value from EEPROM
FB51	READ FLOATING POINT Value from EEPROM
FB46	WRITE BYTE Value to EEPROM
FB48	WRITE INT Value to EEPROM
FB50	WRITE UINT Value to EEPROM
FB52	WRITE FLOATING POINT Value to EEPROM

11.15 Data Type Conversion Functions.

Reference	Description
FB150	CONVERT INT to FLOATING POINT
FB151	CONVERT FLOATING POINT to INT
FB152	RANGE TRANSFORM INT/ FLOATING POINT
FB153	Binary to BCD
FB154	2 Bytes to BCD Byte
FB155	BCD TO Binary Byte
FB156	BCD to 2 Bytes

11.16 Sequence Control Functions.

Reference	Description
FB157	SEQUENCE CONTROLLER
FB158	SEQUENCE STEP HEAD
FB159	SEQUENCE STEP TAIL

11.17 Control Functions.

Reference	Description
FB66	INCREMENT/DECREMENT RATE LIMITER for INT Values
FB67	INC/DEC RAMP CONTROLLER
FB173	FILTER, (very) LOW PASS for INT Values
FB174	FILTER, LOW PASS for INT Values
FB175	LEAD FUNCTION for INT Values

FB176	LAG FUNCTION for INT Values
FB177	DEAD-TIME FUNCTION for INT Values
FB181	PID CONTROLLER (velocity algorithm)

11.18 Constants.

Reference	**Description**
FB195	DEFINE FLOATING POINT Constant
FB196	DEFINE UINT Constant
FB197	DEFINE INT Constant
FB198	DEFINE BYTE Constant
FB210	DEFINE BOOLEAN Constant (System Resource)

11.19 PIC® Device Functions.

Reference	**Description**
FB189	SHOW MICROCONTROLLER DEVICE ID
FB190	SFR BYTE Read
FB191	SFR BIT Read
FB192	SFR BYTE Write
FB193	SFR BIT Write

11.20 Code-Page Functions.

Reference	**Description**
FB200	SIGNAL READ Connector (read signal from another, or same, Code Page)
FB201	SIGNAL WRITE Connector (write signal for use on another, or same, Code Page)
FB202	TEXT COMMENT BOX
FB203	SIGNAL WRITE Connector (for Sequence Output Signals)

12.0 Distribution License.

Users of VPS_P18 must keep in mind that it is only to be used for non-commercial purposes. The package contains certain MICROCHIP proprietary development systems and software code and is distributed under license from MICROCHIP. In this regard please note the following:

Function Blocks, closing the divide between requirement and solution.
Happy PicKing.
Lourens