

StampPlot Pro Version 3.9 (Version 4 Beta)

Primer

June 24th, 2011

The screenshot displays the StampPlot Pro V. 3.9 - Version 4 Beta software interface. The window title is "StampPlot - V4 Beta - SelmaWare Solutions - StampPlot Pro V3.9 ** Unregistered **". The menu bar includes File, Macros, Logging, Plot, Axis, View, Register, Defined, and Help. The toolbar contains various icons for file operations and plotting. The main area is divided into several sections:

- Alarming Meters:** 2 Fully Adjustable Alarming Meters, 5 Channel Selectable Plotting Colors, Full User Interface.
- Standard Plot:** Selectable Plotting Colors, Channel Values, Full user Interface.
- Standard Plot - High Speed:** High Speed Plotting, Channel Values, Full User Interface.
- Four Dial Meters:** High Speed Plotting, Fully Adjustable Alarming Meters, Full User Interface.

**** StampPlot Pro V. 3.9 - Version 4 Beta **** New Meters and Object

- X-Y Plotting:** 3 Graphs Based on sets of X-Y Data, Channel Values, Full User Interface.
- Five Gauges:** 5 Fully Adjustable Alarming Gauges, Min/Max/Ave Values, Full User Interface.
- 10 Bars:** Plots 10 Channels as Bars, Selectable Bar Colors, Full User Interface.
- Digital and Interfacing:** Demo for Digital Data, 8 Digital Input Indicators, 8 Digital Output Selectors, Slider and Gamepad Interface.

At the bottom, there is a status bar showing the time 18:58:44 and four buttons: View Primer PDF, Run V3 Plot Select, and Run No-Frills Plot.

A.	About this Primer	3
B.	What is StampPlot Pro?	3
C.	Plotting Analog & Digital Data, Messages & the Immediate/Debug Window	4
D.	Using the Selectable Interfaces.....	7
E.	Control Instructions, Math, Macro Values & Macros Control Instructions	9
F.	Building an Interface	11
G.	 New Objects and Operations.....	14
H.	 Interactive Control	17
I.	Conclusion.....	18

A. About this Primer

Depending on your level of need, StampPlot Pro can be a fairly simple program to use or a fairly complex program full of options and means to accomplishing tasks. This primer is meant to provide the very basics for new users, to highlight some additions in this release, and to illustrate some common “programming” uses. StampPlot is feature-rich, and as such can be a little intimidating, but basic use for the general user is fairly simple. For much more detail on subjects mentioned here, or to explore other features, please see the help files and the other tutorials posted on our download page for StampPlot at www.stampplot.com (www.selmaware.com). Additionally, we have a yahoo group that you may use for questions.

The help files and other tutorial primers are a little dated and have not been updated well for changes made in StampPlot over the years. This version being called Version 4 Beta due to the amount of changes while we proceed with a few more changes and updating the documentation for an actual Version 4 release some time in the future (2012 most likely).

B. What is StampPlot Pro?

If you’re not familiar with StampPlot Pro as of yet, it is a data acquisition and control program for microcontrollers, allowing plotting, virtual metering and instrument, and control. The true potentation of the software lies in being able to program the hardware (the microcontroller) to interface with user-developed interfaces (called macros).

While the software was developed with the BASIC Stamp in mind, and the materials focus on that controller, it can be used with ANY controller or device capable of sending serial asynchronous data at various Baud rates. The BASIC Stamp® using DEBUG statements, Arduino® and other C-based languages using print or printf statements, PICs, Atmels, etc. The BASIC Stamp’s PBASIC is our default for examples as many are familiar with it. Using those examples a programmer should be able to structure code for their flavor of language.

Connecting to Hardware

StampPlot accepts data through a serial port. In the “old days” this was 9 or 25 pin COM port, when PC’s had 1 or 2. Through the advent of USB, many devices still use a virtual COM port for communications. Due to some legacy software, *StampPlot ONLY accepts up to COM 15*. If you have a device assigned to higher value, you can use Windows’ Device Manage to change the COM number of the device to which you are trying to connect.

StampPlot typically accepts ASCII data from a controller, that’s number, letter, symbols, and control codes sent as ASCII code (commonly called strings). A value such as 105 is sent to StampPlot as characters “1” “0” “5”, as opposed to a binary value of 105 (which with some configuration changes, StampPlot could also accept).

Data Sent to StampPlot

Through the ASCII data send to StampPlot, the following can be performed:

- Multiple channels of analog data can be plotted.
- Multiple channels of digital data can be plotted (on/off traces).
- StampPlot can may be configured using instructions, such as plot scales, and nearly every other aspect of the software.
- Controls (objects) on the interface can be update directly from the controller, such as meters and information boxes.
- Data from controls on the interface may be sent back to controller for interactive control and reading the interface controls.
- Graphics, such as lines and circles may be drawn.
- An entire interface could be configured from the controller itself by sending object instructions to build the interface.

NOTE: Under normal operation (when not in binary mode), ALL data and instructions sent to StampPlot MUST end with a Carriage Return (CR or ASCII 13). This informs StampPlot that a line of data has been received and is ready to be processed.

C. Plotting Analog & Digital Data, Messages & the Immediate/Debug Window

Stamp accepts one or more values separated by commas for plotting. While it can accept up to 100 values at once, it only used the first 10 for plotting. Other values can be accessed as macro values discussed later. A line starting with a value is treated as analog data to be plotted.

It's always a great idea to monitor your DEBUG window or console window of your software to ensure the format looks correct before connecting to StampPlot. Only one software package can connect to a COM port (or USB device) at any one time, so be sure to close your debug/console windows before connect with StampPlot, and disconnecting StampPlot before downloading new code.

Single Analog Value

To send one value for plotting, simply send a single ASCII value ending with a carriage return from your controller:

105 & CR

For example, from the BASIC Stamp, to plot a value:

DEBUG DEC Value, CR

Other controllers have similar means, such as from the Arduino:

Serial.println(value); 'println sends data plus a line feed with carriage return

Multiple Analog Values

Multiple values need to be separated with comma for each to plotted as a separate channel:

105,90,25 & CR

For example, from the BASIC Stamp:

```
DEBUG DEC Val1, “,”;      ‘ comma between value included!  
    DEC Val2, ”,”;      ‘  
    DEC Val3, CR        ‘ end with carriage return
```

Using the Arduino:

```
Serial.print(Val1);      ‘ print with no linefeed  
Serial.print(“,”);      ‘ separating comma  
Serial.print(Val2);  
Serial.print(“,”);  
Serial.println(Val3);    ‘ print with line feed
```

Sending Data for Digital Traces

A line of data from the controller which start with % is treated as binary data to be plotted as digital traces. It’s important to ensure the same number of 1’s and 0’s are always sent (that leading zeros aren’t chopped off by the controller). For example, the following would plot 8 lines of digital traces:

%10110010 & CR

From the BASIC Stamp:

```
DEBUG IBIN8 DigVal, CR   ‘ send binary indicated (%) for 8 bits
```

From the Arduino:

```
Serial.print (“%”);      ‘ leading indicator  
Serial.println(digVal, BIN); ‘ line feed and in binary (not sure about leading 0  
                             ‘ issues)
```

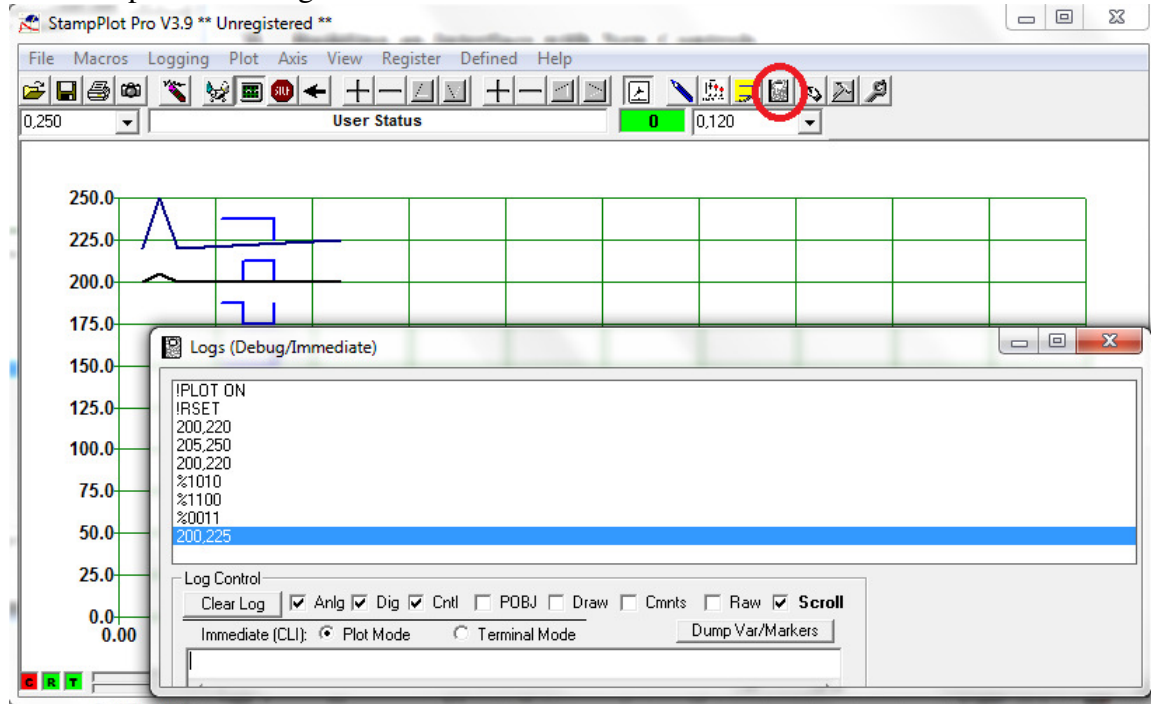
Messages

StampPlot looks for strings that match certain formats, such as starting with a value for analog data, a % for digital data, and the symbols of !, ^, ~ and @ for control and drawing instructions. Anything not matching these criteria will be processed as a message and sent the message window and logged if enabled. For example, sending a string of “Beginning test run” sent from your controller (or the immediate/DEBUG window) will be processed as a string.

Immediate/Debug Window

StampPlot Immediate/Debug window can be beneficial when testing or analyzing problems. The text entry, values and instructions can be tested, and various processes of StampPlot can be monitored.

- Open the Debug/Immediate Window:



- Select to monitor Analog (Anlg), Digital (Dig) and Control instructions (Cntl).
- Enter **!PLOT ON** to enable plotting.
- Enter **!RSET** to reset the plot.
- Enter one or more comma-separated values, such as **200,205** then **210.2,225.5**, etc.
- Enter digital values, such as **%1010**, then **%1100**, etc.

The data you enter in this window is identical to what may be sent from your controller for operation, properly formatted by your code. And this may be used to monitor incoming data if things don't go the way you anticipate.

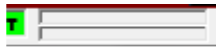
Sending analog and digital data correctly for StampPlot covers the majority of needs for many users.

D. Using the Selectable Interfaces

To ease use for basic needs, an assortment of interfaces have been designed to select from. Basic steps in using:

- Have your microcontroller ready to send analog an/or digital data.
- Select an Interface by clicking on it.
- Select the correct COM port in the lower left drop-down box.
- Select **Connect-Plot**
- Watch your data be plotted and metered.
- Test out the various buttons.
- Data Logging allows data and message to be logged to text files for importing into spreadsheets or other uses.

The first 7 interface provided plot and meter analog data, though digital data maybe sent as well (such as the state of a heater for an incubator). Several are labeled as “High Speed” – These have little processing overhead and can accommodate faster data. The others require data a little slower. A good indication is the queue indicator (bottom bar) which displays how much data is waiting to be processed. If it begins to fill, you will want to slow down your data. Above the Queue indicator is the data point indicator – how full the point memory is. All the interfaces are set up to “flush” old data to keep the memory from topping out – but when redrawing old, flushed data will not be redrawn. Generally, give StampPlot 20 to 100mS to process data.



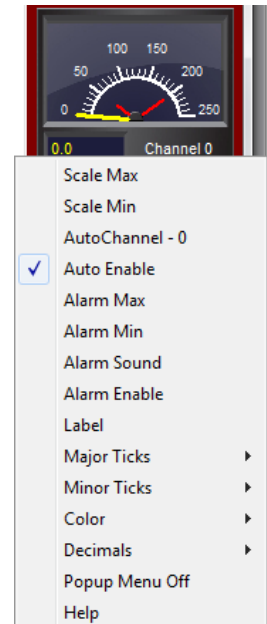
Some Common Interface Controls



Starting from the left:

- The **question mark** checks for available COM ports (serial connections) that you may connect to. If your device is not listed, be sure it has a port number less than 16 and is not being used by another application.
- **Port number list and Baud rate.** Ensure the Baud rate StampPlot and your controller match or you will get garbage.
- **Reset Plot:** Clears all data and returns to time 0.
- **Reset Axis:** Returns the X and Y axis to the default for the interface.
- **Auto Y-Axis:** Sets the Y-Axis based on received data since last reset.
- **Y adjust group:** Adjust scale of Y-Axis
- **Time Inc and Time Dec:** Adjust X-Axis (normally time).
- **Real Time:** Sets X-Axis to show real time instead of time into plot.
- **Shift Max:** When enabled (green), cause the plot to shift when the maximum time has been reached.
- **Stop Max:** When shifting is off, can be used to stop plotting when the time reaches maximum.

- **Log to File group:** Names, enables, and opens the data and message log. The data log is in the formation of: date, time, seconds, digital data, analog value 1, analog value2, etc.
- Some interfaces have **color selectors**, click these to select the color for various channels of data. The ability to select colors slows processing.
- **Width:** Sets the width of the plot line.
- **Snapshot:** Takes an image of the interface, names based on the interface name and appends date & time.
- **View:** Opens the last snapshot for viewing.
- **Folder:** Opens the folder where logs and snapshots are stored.
- **Save Settings:** Should you adjust the interface, such as meter ranges, colors, file names, etc and wish to recall it later, use this button to save your settings to the Windows registry. Data is stored based on the title of your plot.
If you wish, you can change the title by using the Immediate window and typing !TITL myplot (or whatever).
- **Load Settings:** Reloads your settings from the registry based on the title.
- **Plot Select:** Returns to the plot selection interface.
- **Meters:** The ranges of the meters, alarm points, tick marks, labels, colors and other features of the meter may be changed by right-Clicking a meter and choosing what to change (currently Help is not supported).
- The **Alarm On** buttons on the interface will enable the alarm features of the meters – should the value go above or below the max or min setpoints, a sound file will be played. Also, when in an alarm state, the meter label will turn red. When not alarming, the title will be green. If not enable, the meter label will be the default color.



More on meters will be covered in the overview of the new controls.

E. Control Instructions, Math, Macro Values & Macros

Control Instructions

As demonstrated earlier, nearly all aspects of StampPlot can be controlled using control instructions. From selecting a COM port and connecting, scaling the plot, enabling plotting to placing objects such as meters on the plot and controlling them. Control instructions are 4-letter abbreviations beginning with a !, such as:

!PLOT ON	Enable plotting
!SPAN 0,100	Scale Y axis
!RSET	Reset Plot
!PPER 50,50	Set plot percentage of interface
!STAT Hello!	Place data in the status box above the plot
!DEBUG Hello!	Show data (debug) in Immediate/DEBUG window

These instructions can be entered in the Immediate/DEBUG window or sent as control instructions from your controller allowing full configuration of the interface from your hardware. They are also used in macros (interface configuration files) to create and control the StampPlot interface

When you don't want the data sent to be plotted automatically, or if you want more control such as defining the color, the !ACHN (analog channel) instruction can be used. Automatic plotting of analog data can be done by issuing the !USEA ON instruction (use analog data for macro only).

!ACHN Channel (0-9), Value, Color
such as:

!ACHN 1, 500, (RED). Plot on channel 1 the value of 500 in red.

Of course this can be properly formatted to be sent from the controller as a string.

Please see the Control Instructions Summary for a listing of instructions – though this file needs updating in the future for recent additions

Macro Values

Data may be retrieved from StampPlot for use or monitoring by accessing data as macro values. These are typically accessed by placing a control name or other parameter in parenthesis (). These may be used in instructions and in math operations.

Examples to show in the Immediate/DEBUG window:

!DEBUG (PTIME)	Shows time into plot
!DEBUG (RTIME)	Shows real time (HH:MM:SS)
!DEBUG (TMAX)	Shows maximum time of plot

A complete listing of macro values is in the help file math summary.

When analog data arrives, it is parsed into macro values as separate channels, called AINVAL0 to AINVAL99 – Analog input values, channels 0 to 99.

For example, if data received were 10,20,30

!DEBUG (AINVAL0) would show 10

!DEBUG (AINVAL1) would show 20

And so on.

Digital data is similar. Receiving %1101

!DEBUG (DIN) would show the binary value of 1101.

!DEBUG (BIT0) would return bit 0 value (1)

!DEBUG (BIT1) would return bit 1 value (0)

And so on

Object values and properties can also be accessed using macro values. For example, a meter named met1 indicating a value of 50 can be accessed:

!DEBUG (met1) would return the current value of the meter.

!DEBUG (met1.amax) would return alarm maximum of met1.

The same principle can be applied to all object that the user may interface with, such as sliders, text boxes, drop down boxes, etc.

Math Operations

StampPlot has the ability to perform math and other operations on data. Math is signified by using brackets [], and EACH PARAMETER MUST BE SEPARATED BY SPACES. The math operations can be sent from the controller or used by your interface instructions.

!DEBUG [5 / 2]

Show math result of 5 / 2

!DEBUG [(AINVAL0) / 100]

Show analog value0 divided by 100

!POBJ met1=[(AINVAL0) / 100]

Update met0 with AINVAL0 divided by 100

!DEBUG [1 / 3 + 10 SIN FORMAT 0.00]

Show result of 1/3+10 formatted for Decimals

All math is operated left to right, without precedence unless nested brackets are used:

!DEBUG [1 / [3 + 10] FORMAT 0.00]

Note: The help files have not fully integrated the simplified math shown from earlier versions.

The microcontroller can also integrate the math so that StampPlot processed the values using math prior to use by sending properly formatted data. For example, BASIC Stamp code for the following operation:

[Temperature * 500 / 255] to scale ADC data from 0 to 500 degrees:

DEBUG “[“, DEC Temperature, “ * 500 / 255]”, CR

StampPlot also has an IF-Then statement which can be useful, such as if a value above a certain value, then perform an operation:

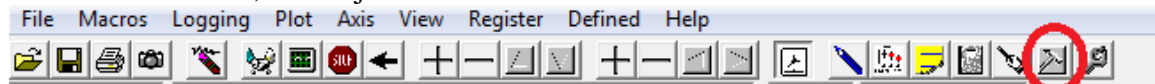
‘ sound bell and reset if value exceeds 100:

```
!IFTH (AINVAL0),>,100,!BELL(CR)!RSET
```

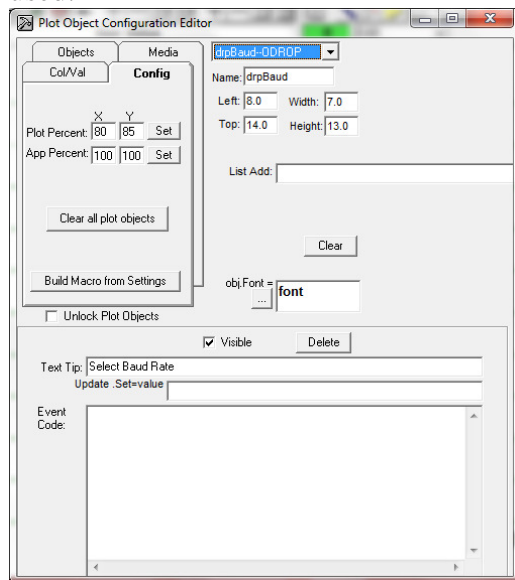
F. Building an Interface

An interactive user interface uses plot objects (controls) to interface with the data and the user. There exists 2 groups – those natively built into StampPlot, and additional controls made for StampPlot as OCX files. The new added controls provide easier use and functionality in many cases.

To use the controls, the Object Editor is used.



The first step, when starting from a no-frills plot, is to reduce the plotting area to allow a background area on which to place the controls. The !PPER instruction may also be used.



Once complete, controls can be dragged and dropped on a clear background area. Once on the interface, they can be positioned and sized using the boxes for “Left”, “Width”, etc. The simplest way to move controls is to place your cursor in the box and use the arrow keys on your keyboard to move the control around or resize (width/height). Hold down the shift key while using arrow keys for small changes.

While this primer will not go through extensive examples (there is a developer tutorial primer on our downloads page – though dated – it can help a lot) we will discuss some basics.

- **Name** your controller appropriately for use, such as metPressure (a meter for pressure). You may need to access the value it holds, such as

`!POBJ txtPressure = (metPressure)` to set an object called `txtPressure` to the value of `metPressure`.

- **Unlock Plot Objects** allows you drag them to another blank area on the screen.
- **Text Tip** provides text when the mouse is above the control.
- **Update .Set** can be used to “automatically” update a control object to a value. For example, when analog data arrives, you want a control to show value of analog channel 0, `AINVAL0`). In this box would place `(AINVAL0)`

It can also do math, such as a value of voltage (`AINVAL0`) times the current received (`AINVAL1`) for show power.
`[(AINVAL0) * (AINVAL1)]`

The objects update values are **ONLY** processed when a **!POBJ Update** is issued. This can be sent from the controller, or you can use an object in the drop-down list called **oAnalog** and place the `!POBJ Update` instruction in the event code box there. This object is ran anytime analog data arrives. A timer object (in the list) may also be used to do the updates periodically.

- Event Code is where you would place StampPlot Instructions to be performed when a control is triggered – such as a button being clicked or a meter exceeding a setpoint. For example, if you have a button labeled “Reset Plot”, when the user click it, it will run the code listed, such as:

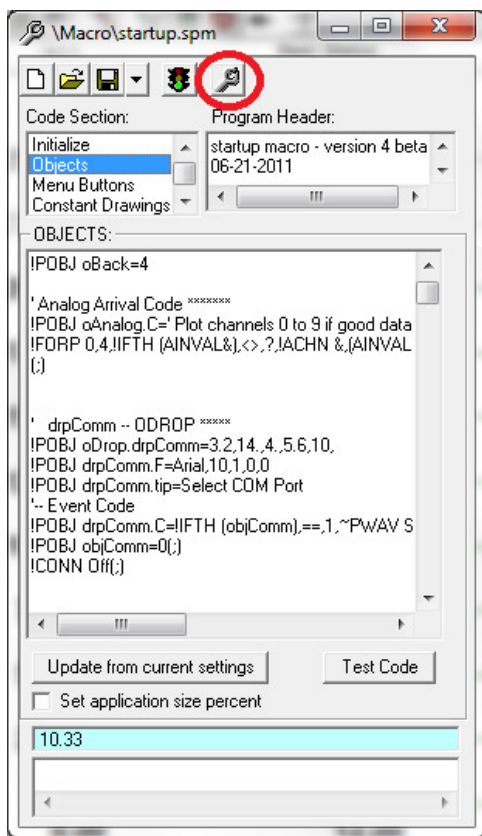
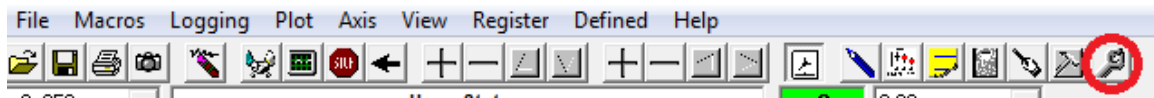
```
!RSET
‘ Always reset twice due to a small bug
!RSET
‘ Scale plot
!SPAN 0,100
‘ Sound the computers bell.
!BELL
```

Notice the comments may be used – I’m bad at commenting mine, sorry!

- The Object editor for controls may be opened by using holding down the shift-key and right-click an object. Some objects, such as text boxes and drop-down boxes don’t open too smoothly.
- Use the Delete button sparingly – it sometimes causes issues.
- When used in an objects code `((ME))` returns the value of the current object. An example may be a slider where when you adjust it you want a textbox to update with a value, so in the sliders code you enter: (note: `!O` is shorthand for `!POBJ`):
`!O txtValue =((ME))`

Building an Interface Macro

Once you have an interface working the way you want, you can build a macro file defining all aspects of your interface. Open the Macro Builder.

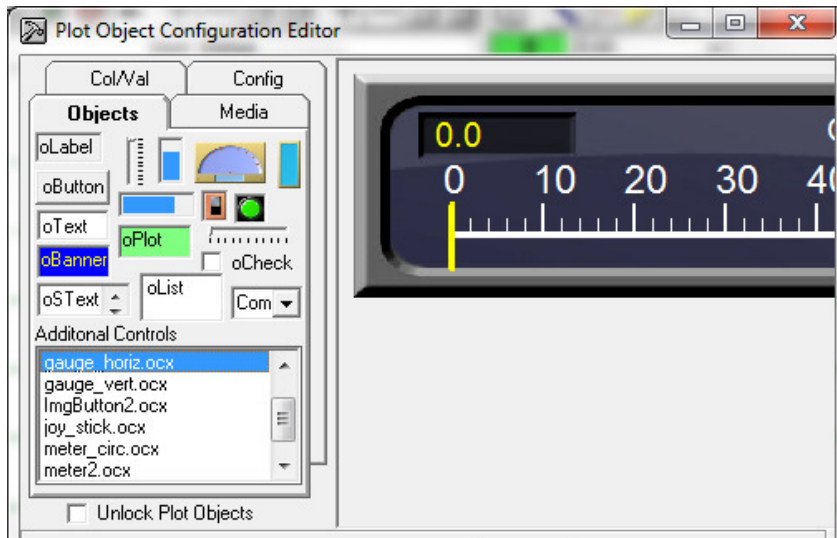


Hit the Build button and the a configuration macro (StampPlot Macro) is built from your settings. Save the file and the macro is ready to be opened and test. The Stoplight icon can be used to open run the macro, or simply double-click the file to open.

Many objects use graphics, such as image buttons. If you make your own graphics or use other sound files that are supplied with StampPlot (in the media directory), place all files and in a common directory and build the macro to that directory – say yes when asked to build to the directory – this will update the path to the files. This will make your macro and files portable to another computer or location.

G. New Objects and Operations

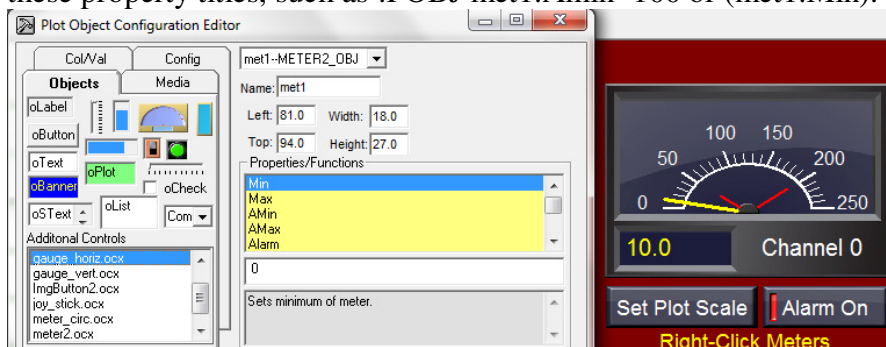
This primer is a quick overview, but for current users and developers we need to highlight some of the new additions and updates for StampPlot. Multiple new meter styles and a new image button have been added as add-on controls. These new controls are actually much better developed for use than the standard built-in controls, and are listed in a box below the standard controls.



New Meters and Gauges

Gauge_Horiz, Gauge_vert, Meter_circ and Meter2 all share common code and functionality. Here are some highlights and use.

- The properties of the meters and gauges have a better defined property window, with choices, selectors and help. When accessing properties of the meter, use these property titles, such as !POBJ met1.Amin=100 or (met1.Min).



- A right-click on the objects brings up a pop-up menu with similar functionality and choices. Some specifics on use:
 - **Auto Channel:** Use to select the analog channel which to update with, 0 to 99. Simply enter the channel value, such as 1.

This value may also be a macro value or other object property, such as AINVAL20. Do NOT use parenthesis or use math.

- **Auto Enable:** Enables auto-updating. The meter updates in the background.
- **Alarm Sound:** Access the media directory to chose a sound file.
- **Alarm Enable:** As mentioned enabled, the meter color will be either green or red based on the status, and the alarm will sound when beyond a limit.
- **Label** is the title on the gauge.
- The number of tick marks, major and minor, are adjustable.
- **Color** will allow you to choose one of the designed meter color schemes.
- **Decimals** will format the displayed value.
- **Popup Menu Off** will hide the popup (in case you want it for 3rd party use that you don't want messing with the settings).

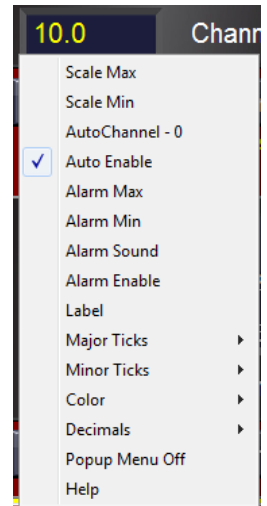


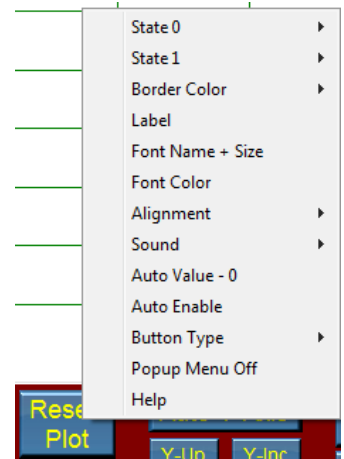
Image button 2 (imgButton2)

The new image button is greatly enhanced version of the image button. Here are some of it's features:

- Can have a defined image or color for up (image state 0) or down (image state 1).
- Can have a defined label.
- Allows auto updating of digital or other values.
- Built-in sound choice.
- Object editor or popup menu adjustable.
- Three types of operation: On/Off, Momentary or static buttons.
- Use an auto-updating label for data.

Some menu choices:

- **State 0 and State 1:** Selects either an image or a color for the up state (0) and the down state (1) of the button. Multiple new images have been created for the buttons, in Media\V4 directory, with choices for on/off control indicators. In code this property is called image0 and image1 for both color and images.
- **Border color** sets a border color if desired.
- **Label** sets text on the button.
- **Font Name + Size** controls the font type and size. Note that all font choices (bold, etc) are not yet supported.
- **Alignment** for the position of the text.
- **Sound:** Defines a sound when operated.
- **Auto Value:** When a value, such as 0 to 15, is set, this corresponds to incoming digital data bit state, Bit0 to Bit15. In this mode it will control the state of the button – up or down. Incoming data will active the button to the state define by the bit – 0 or 1.



When other values are entered, such as AINVAL0 or RTIME, the label of the button will be updated with those values. Again, no parenthesis or math is allowed.

- **Auto Enable** allows background updating.
- **Button type** chooses between On-Off (2 state), momentary (pops back up), and Static – not clickable for text only.

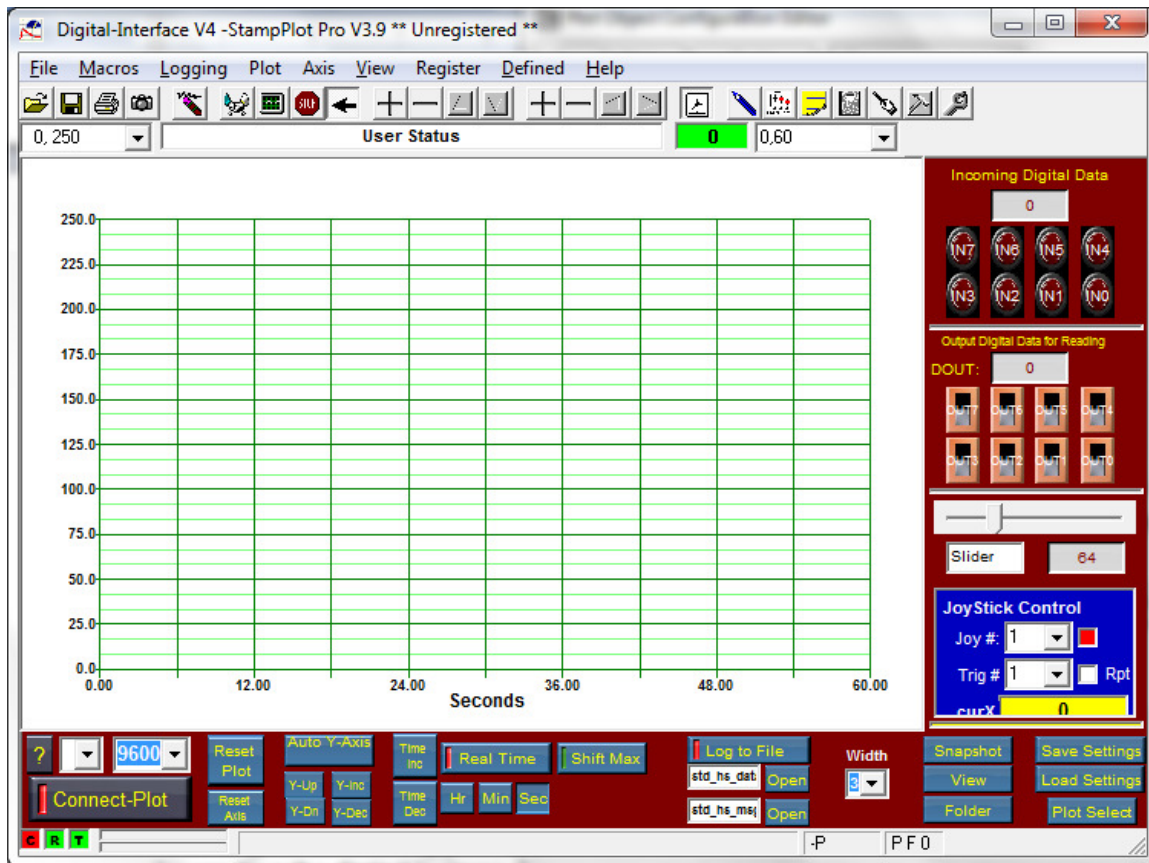
With a little graphics work, some interesting graphics for these buttons can be developed for your unique interfaces.

Joystick/Game Pad Control

The joystick/gamepad was developed for a school participating in underwater ROV competitions. It allowed control via a joystick and monitoring of the system with StampPlot. We'll talk about interactive control in the next part, but it allows the joysticks, pad, and buttons of a USB gamepad or joystick to be monitored and read by the controller via StampPlot.

H. Interactive Control

The Digital-Interfacing plot allows monitoring of incoming digital data, and the control by reading the digital buttons, the analog slider and the joystick/gamepad control. We will discuss methods to read controls on the interface from your controller.



As incoming digital data arrive (%10100010), the LEDs IN0 to IN7 will be updated to reflect each bits status.

To read the interface, a !READ command is issued by the controller, StampPlot returns the value serially to the controller. In pseudo-code from the controller side:

- Issue !READ (DOUT) to request value of control DOUT
- Accept the incoming serial data from StampPlot

For the BASIC Stamp:

```
DEBUG “!READ (DOUT)”,CR      ‘ Request data
DEBUGIN DEC dataIn          ‘ Accept data
PAUSE 20                     ‘ allow echo to clear
```

Note: Using SERIN vs DEBUGIN allows the user to use a timeout value in case things go wrong to prevent lockup on the BASIC Stamp code.

We have done similar with the Arduino, but the code was a little messy – maybe you can come up with good code for us?

StampPlot timers (oTimer) may be set up to send predetermined, formatted data periodically as well.

On the interface, the following may be read:

- Switches **OUT0** to **OUT7** as bit values (0 to 1).
- **DOUT** may be read, which hold the values of bits OUT0 to OUT7 converted to an ADC value (ADC is a math command:
`!POBJ DOUT= [(OUT7)(OUT6)(OUT5)(OUT4)(OUT3)(OUT2)(OUT1) ADC]`)
- **Slider**, which is set for a value range of 0 to 255.
- The joystick/gamepad may be read as well if you have one connected, including:
 - **Joy.CurX** - X position of joystick 1
 - **Joy.CuyY** – Y position of joystick 1
 - **Joy.curX2** – X position of joystick 2
 - **Joy.curY2** – Y position of joystick 2
 - **Joy.curPOV** – value of the thumbpad
 - **Joy.btn 0** to **Joy.btn 15** – bit value (0/1) of buttons 0 to 15. When the buttons are pressed, the corresponding button color is highlighted.
- The event code for the control can be controlled to be from any of the buttons.
- A choice can be made whether holding the button repeats the event code for the control

Math can be applied when reading the interface as well (in fact, it could simply be used as a math coprocessor without accessing any actual data in the screen):

```
!READ [(Slider) * (joy.CurX)]
```

Other Things of Note

An object named **objComm** is automatically mapped to connect/disconnect in that a connection error will change the button state. It will not be affect by manual connecting/disconnecting.

A drop down list or list box named **drpComm** will be automatically updated with a list of available COM ports when the interface is opened.

I. Conclusion

StampPlot Pro is an extremely versatile program for monitoring and control. Hopefully this tutorial gave a flavor to introduce you to some concepts and new additions. Everyone has unique uses, and we encourage you to view the help files and tutorials on our downloads page. Also, please use our yahoo group for more direct support. While we support Email support, we encourage only licensed users to take advantage of this.

Thank you and Enjoy!

Martin Hebel

SelmaWare Solutions