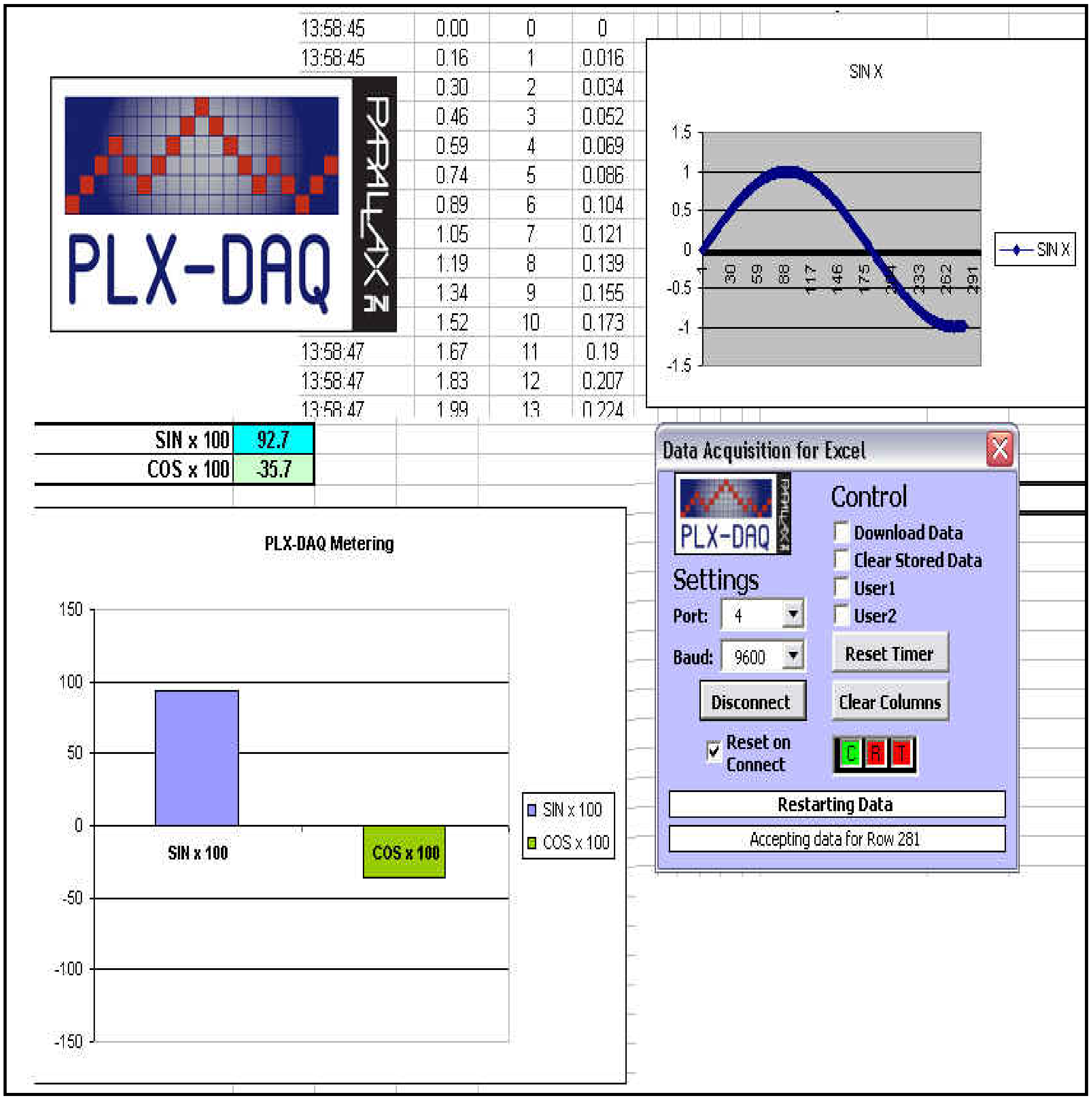
Parallax Microcontroller Data Acquisition for Excel



13:58:45	0.00	0	0
13:58:45	0.16	1	0.016
	0.30	2	0.034
	0.46	3	0.052
	0.59	4	0.069
	0.74	5	0.086
	0.89	6	0.104
	1.05	7	0.121
	1.19	8	0.139
	1.34	9	0.155
	1.52	10	0.173
13:58:47	1.67	11	0.19
13:58:47	1.83	12	0.207
13:58:47	1.99	13	0.224

SIN x 100	92.7
COS x 100	-35.7

Port: 4
Baud: 9600
[✓] Reset on Connect

Restarting Data
Accepting data for Row 281

For the BASIC Stamp 2, SX and Propeller Controllers

www.parallax.com

Features:

- Plot or graph data as it arrives in real-time using Microsoft Excel®
- Record up to 26 columns of data
- Mark data with real-time (hh:mm:ss) or seconds since reset.
- Read/Write any cell on a worksheet
- Read/Set any of 4 checkboxes on control interface
- Example code for the BS2, SX (SX/B) and Propeller available.
- Baud rates up to 128K

Copyrights

[Home] [Copyrights]
[Basic Principles] [Getting Started]
[Control Directives] [Plotting Example]
[Interactive Metering Example]

Copyrights and Trademarks

Copyright © 2007 by Parallax, Inc. All rights reserved.

PLX-DAQ is a trademark of Parallax, Inc. BASIC Stamp is a registered trademark of Parallax, Inc. Excel, Visual Basic and Windows are registered trademarks of Microsoft Corporation. Other brand and product names are trademarks or registered trademarks of their respective holders.

BS2 and Propeller code provided by Martin Hebel, Southern Illinois University, Electronic Systems Technolgoies.

SX/B Example code provided by Terry Hitt of Hitt Consulting.

Disclaimer of Liability

Parallax, Inc., is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, nor any costs of recovering, reprogramming, or reproducing any data stored in or used with Parallax products.

Distribution

The PLX-DAQ installation setup may be freely distributed AS-IS.

Modified Excel worksheets may be freely distributed to other users of PLX-DAQ.

Modifications to the PLX-DAQ macro is authorized for personal use only.

Use of the serial communications Active-X Control (selmaDAQ_V2.ocx) included with this package may not be distributed in other packages or uses without the consent of SelmaWare Solutions. Please contact support@selmaware.com.

Basic Principles [Home]

PLX-DAQ Basic Principles

General
Data, in specific formats, is sent from the controller to the computer's serial port. A Visual Basic for Applications (VBA) macro containing a serial port control is used in Excel to accept data from the serial port, analyze it, place the data in the spreadsheet or perform other actions. Directives are used to inform PLX-DAQ of what action is to be taken.

Directives
PLX-DAQ analyzes incoming data strings from the BASIC Stamp for action. Strings begin with a directive informing PLX-DAQ of what action to take. Most all controllers have a means to send serial data to the PC. The data sent must be formatted properly to be understood by PLX-DAQ.

- All directives are in CAPITAL letters, and some are followed by comma-separated data. Each string MUST end in a carriage return (CR).
- Strings not beginning with directives will be ignored.
- Strings containing ASCII characters < 10 or > 200 will not be processed and indicated as an error.
- Example directive:
  DATA,123,345,567
  Will place the 3 values in the 1st 3 cells of the next row.

Plotting or Metering
Beyond collecting data, PLX-DAQ may be used for real-time plotting or metering. Using the DATA directive, data may be plotted using graphing features of Excel as data fills rows. Through the used of the CELL,SET directives, code may directly update cells allowing real-time metering using graphs in Excel.

Serial Communications
The computer serial COM ports are used to communicate with the controller. PLX-DAQ supports Baud rates up to 128,000. If you are using a USB device for

communications, many of these devices create a virtual COM port which may be accessed as regular COM port. Your programming software may tell you the port it is programming through, or you can use Device Manager of Windows to view the available ports. *Only COM Port 1 - 15 are supported by this software.*

One Port, One Application
Only ONE application can have control of a serial port at any one time. If you use the same serial for programming and communications, you will need to disconnect PLX-DAQ prior to programming and close any terminal window your programming software may use, such as a DEBUG window.

Speed Limitations
Serial Data: Serial data is transmitted one-bit at a time, including Start and Stop bits. 9600 refers to the ability to send a byte at 9600 bits per second. One byte (or character), plus start and stop bits, is 10 bits long. At 9600 baud this would take 1/9600 * 10 = 1.04 mS or .00104 seconds. With a string such as DATA,65,66, which is 10 characters, plus a CR or carriage return, for 11 bytes. 1.04 * 11 = 11.44mS or .0114 seconds.

PLX-DAQ: This application can only accept and use the data as fast as the program can run. On slow computers it may not be able to process the data fast enough to maintain real-time. Variables in maintaining real-time include the speed of the computer (223MHz or 1.2GHz?), the rate at which data is being sent from the BASIC Stamp (once a second or every 10mS?) and how Excel is using the data (just placing in rows or graphing and performing calculations while the data is arriving?). If you require high speed data in real time and wish to graph it, it is best to place the graph on a second Excel worksheet and not view it. Viewing a graph will slow the processing time considerably. PLX-DAQ has a 5000 character buffer in the event data arrives faster than it can be processed by Excel.

Getting Started [Home]

Getting Started with PLX-DAQ

System Requirements

- Windows 98 or higher.
- Microsoft Office 2000 or higher.
- Communications port for programming and data communications.
- Microcontroller hardware and editor.

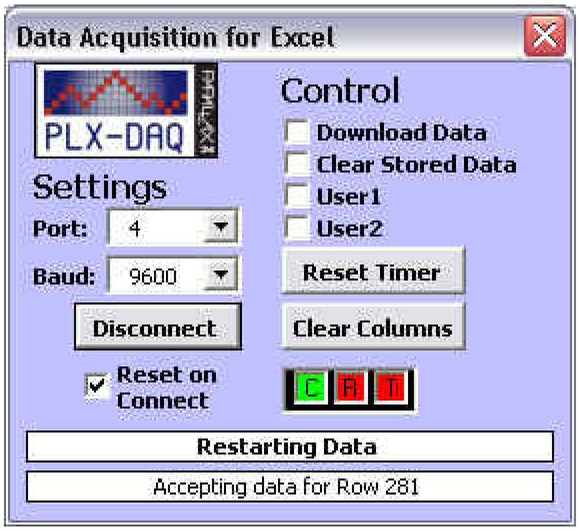
Excel Macros and Security
Due to the malicious nature of some VBA macros, Microsoft has security features associated with the use of macros.

- Upon starting the spreadsheet, Excel may ask if you wish to allow the macro to run. For

  PLX-DAQ to operate you will need to allow it.

- Excel may be set for "High" security and Excel will not ask about running macros, nor will it allow macros to be run. To switch to "Medium" security:
  - From Excel, Select menu item "Tools-->Macro-->Security"
  - Set security to "Medium" to allow Excel to ask about running macros. "Low" is NOT recommended for security reasons.
  - Close Excel and re-open the PLX-DAQ spreadsheet.

PLX-DAQ Interface



- Once the spreadsheet is loaded the interface should open automatically, or pressing

  <u>CTRL-q </u>on the keyboard will open the interface.

- The settings may be used to select the COM port and the Baud rate for the data.
- <u>Reset on Connect </u>will cycle the COM port DTR line.

  With some controllers this will cause the controller to reset.
  This will work with the BASIC Stamp and the

  Propeller (if using the programming port for communications).

- Clicking the <u>Connect</u> button will connect on the specified COM port at the specified baud rate (8 bits, no parity, 1 stop bit). Disconnect by clicking it again.
- The data communication indicators will indicate:
  - Not Connected - Red C
  - Connected - Green C
  - Data Received - Red R
  - Data Transmitted - Red T
- Use the <u>Clear Columns</u> button to clear columns A-J (all rows except row

  1) or the number of columns data was placed in (up to 26).

- Checkboxes are used for interactivity.

  The controller can read the checkboxes using directives.

  - Download Data can be used to dump stored

    data for retrieval dependent on user's controller code.

  - Clear Stored Data can be used to clear

    any stored data dependent on user's controller code.

  - User1 and User2 are user defined

    checkboxes and may be labeled by your controller.

- Reset Timer will reset PLX-DAQ's timer

  to 0 which is accessed using the TIMER directive.

Modifying & Saving Your Spreadsheet

- You may add graphs and charts and calculations of your

  PLX-DAQ data to your spreadsheet (see limitations below).

- You may modify the spreadsheet in other ways to use the data as you desire.
- You may save your spreadsheet and re-open it with your changes, or save to other names.
- You may share your modified spreadsheets with other users of

  PLX-DAQ.

Limitations

- PLX-DAQ can accept 26 comma-separated values for use in Excel.
- ***PLX-DAQ will ALWAYS place incoming data on the 1st sheet of your workbook.***

  You may create additional sheets and charts, but PLX-DAQ will use the 1st in the tabbed-list for incoming data.
- If you are plotting high-speed data, it is recommended you place charts and graphs on separate sheets.

  Viewing of charts and graphs as data arrives will dramatically slow the processing of incoming data.

- Sharing of modified PLX-DAQ spreadsheets, such as with special graphs or calculations, is encouraged though the end-user will need to have installed

  PLX-DAQ to use the data acquisition features.

- Distribution of modifications to the PLX-DAQ macro is not authorized under the name of

  PLX-DAQ or other names without consent of participating companies. Please see the copyright notice.

- For simple error checking, PLX-DAQ will indicate an error anytime that a string containing characters < ASCII

  10 or > ASCII 200 is received.

- Values of ASCII 10 (Line Feed) are replaced

  with ASCII 13 (Carriage Return) prior to processing.

Control Directives [Home] [Copyrights]

[Basic Principles] [Getting Started] [Control Directives]
[Plotting Example] [Interactive Metering Example]

PLX-DAQ Directives

PLX-DAQ analyzes incoming data strings from the

controller for action. Strings begin with a directive informing PLX-DAQ of

what action to take.

- All directives are in CAPITAL letters, and

  some are followed by comma-separated data. Each string MUST end in a

  carriage return (CR).

- Strings not beginning with directives will be

  ignored.

- Note that the BASIC Stamp's DEBUG instruction is used in

  the following discussion.

Control and **Data Directives**	**BASIC Stamp** **Example Usage**
DATA, value1, value2,... Stores the data into the next row from column A on.	Up to 26 comma values may be sent. The 1st value will be placed in column A, the 26th in column Z. **DEBUG "DATA,", DEC Val1, ",",** **DEC Val2, CR** Special: The strings **TIME, TIMER** and **DATE** in value positions 1 and 2 will cause PLX-DAQ to replace those strings with the current time (TIME) or the time in seconds since opening or the last timer reset (TIMER), and date when using the DATA directive. **DEBUG "DATA, TIME, TIMER,", DEC** **Val1,",",DEC Val2,CR**
LABEL **,**	**DEBUG "Time, Timer, Value 1, Value 2", CR**

label 1, label 2, label 3,... Labels the columns from A up to Z. Row 1 is used for labels, up to 26 labels may be sent.	
CLEARDATA Clears columns A-J of data from Row 2 on (labels remain in Row 1), or the number of columns to which data was being posted.	**DEBUG "CLEARDATA",CR**
RESETTIMER Resets the when using the TIMER directive to place the time in seconds into the cell.	**DEBUG "RESETTIMER",CR**
MSG, message string Places a message in the PLX-DAQ Controller Message box. Do not use commas in the message string.	**DEBUG "MSG, Starting data run.",CR**

Interactive Directives	**BASIC Stamp** **Example Usage**
ROW,GET Retrieves the last row number data was placed into using the DATA directive.	**DEBUG** **"ROW,GET",CR** **' Request row** **DEBUGIN DEC**

	row **' Accept row value into variable** or **SEROUT 16,84,["ROW,GET",CR]** **' Request Row** **SERIN 16,84,500,Timeout,[DEC row] ' Accept row value with timeout** **Timeout:**
ROW,SET,Value Sets the row number to use for next DATA	**DEBUG** **"ROW,SET,2",CR ' Send next** **data to row 2**
CELL,GET,cellnum Retrieves the value in a specified cell.	**DEBUG** **"CELL,GET,A4",CR** **' Request cell value** **DEBUGIN DEC cellval** **' Accept cell value into variable** or **SEROUT** **16,84,["CELL,GET,A4",CR]** **' Request cell** **SERIN 16,84,500,Timeout,[DEC cellval] ' Accept cell value with** **timeout** **Timeout:** Note: variables and constants may be used for cells in the A-F range by using hexadecimal. **Cell CON $A4** **DEBUG "CELL,GET,", HEX CELL,CR**
CELL,SET,cellnum,cellvalue Sets the specified cell value or text	**DEBUG** **"CELL,SET,B4,Hello!",CR** **DEBUG "CELL,SET,B5,", DEC x, CR** Note: variables and constants may be used for cells in the A-F range by using hexadecimal. See CELL,GET.
DOWNLOAD,GET **STORED,GET** **USER1,GET** **USER2,GET** Reads the status of one of the four checkboxes on the PLX-DAQ interface.	**DEBUG** **"USER1,GET",CR** **' Request checkbox value** **DEBUGIN DEC** **check1**

0 for unchecked 1 for checked.	**' Accept value into variable** **IF check1 = 1 Then ...** or **SEROUT** **16,84,["USER1,GET",CR]** **' Request cell** **SERIN 16,84,500,Timeout,[DEC check1] ' Accept cell value with** **timeout** **Timeout:**
DOWNLOAD,SET, 0 or 1 **STORED,SET, 0 or 1** **USER1,SET, 0 or 1** **USER2,SET, 0 or 1** Sets the status of one of the four checkboxes on the PLX-DAQ interface. 0 for unchecked 1 for checked.	**DEBUG** **"USER1,SET,1",CR** **' Sets checkbox USER1 to 1**
DOWNLOAD,LABEL,text **STORED,LABEL,text** **USER1,LABEL,text** **USER2,LABEL,text** Sets the text of the checkboxes	**DEBUG** **"USER2,LABEL,Check me!",CR ' Sets USER2 checkbox** **label.**

For prior StampDAQ users, the **CMD?, DONE, RESET, DUMPING** directives

are still operational, but the new interactive directives are recommended.

Plotting Example

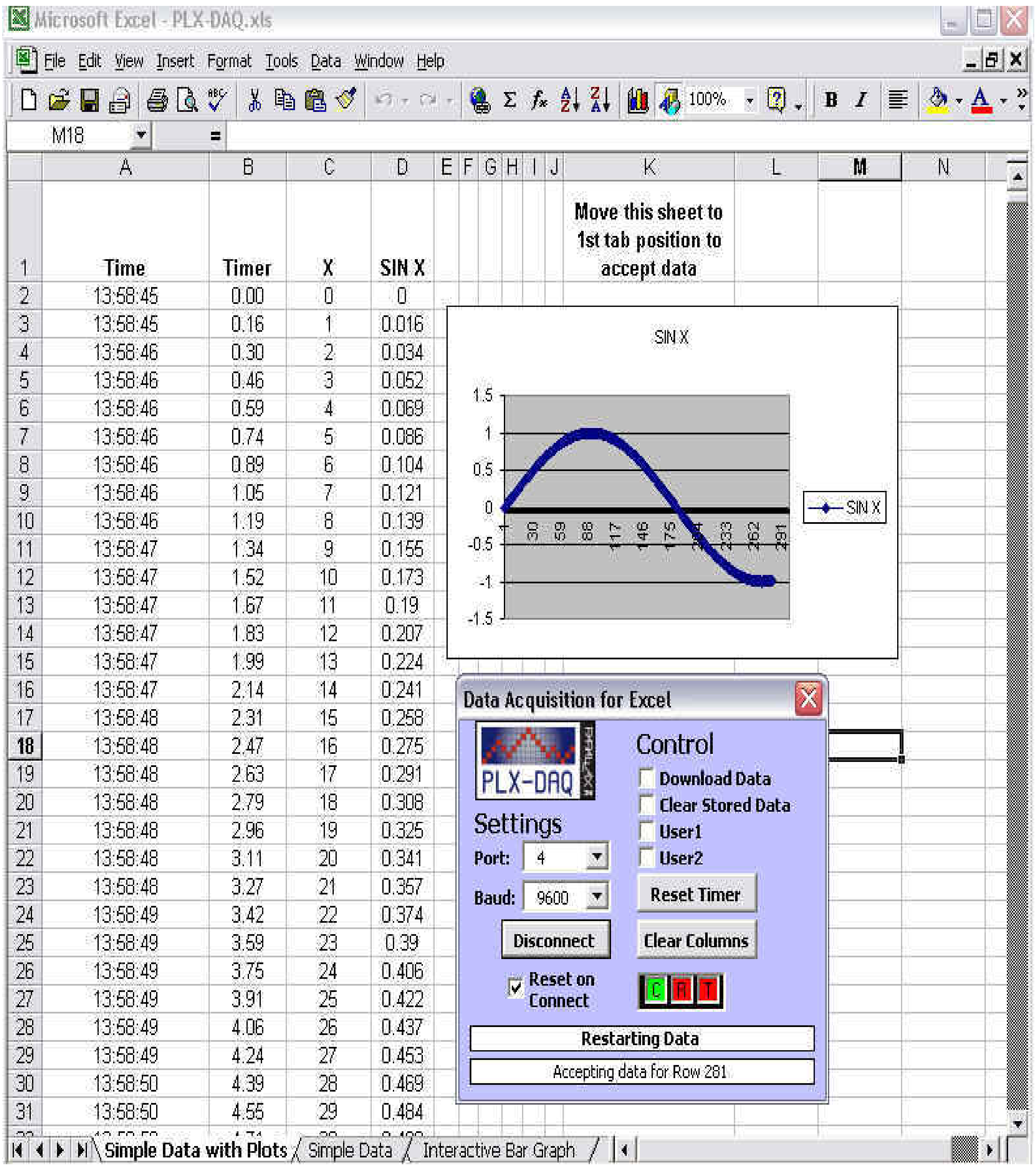
[Home]
[Copyrights] [Basic Principles] [Getting Started]
[Control Directives] [Plotting Example]
[Interactive Metering Example]

Plotting Example

This example uses the "Simple Test" source code and the "Simple Data with Plots" worksheet. Example code for the BS2, SX, and Propeller are provided in a separate download available on Parallax's PLX-DAQ page.

The code performs the following:

- Uses the **DATA** directive to record data of time (**TIME**), time since reset (**TIMER**), and 2 values of a count and SIN of that count value.
- Monitors using **ROW,GET** when the row has exceeded 300.
- Resets the row back to 2 using **ROW,SET**.
- Graphs the data using graphing features of Excel.



	A	B	C	D
1	Time	Timer	X	SIN X
2	13:58:45	0.00	0	0
3	13:58:45	0.16	1	0.016
4	13:58:46	0.30	2	0.034
5	13:58:46	0.46	3	0.052
6	13:58:46	0.59	4	0.069
7	13:58:46	0.74	5	0.086
8	13:58:46	0.89	6	0.104
9	13:58:46	1.05	7	0.121
10	13:58:46	1.19	8	0.139
11	13:58:47	1.34	9	0.155
12	13:58:47	1.52	10	0.173
13	13:58:47	1.67	11	0.19
14	13:58:47	1.83	12	0.207
15	13:58:47	1.99	13	0.224
16	13:58:47	2.14	14	0.241
17	13:58:48	2.31	15	0.258
18	13:58:48	2.47	16	0.275
19	13:58:48	2.63	17	0.291
20	13:58:48	2.79	18	0.308
21	13:58:48	2.96	19	0.325
22	13:58:48	3.11	20	0.341
23	13:58:48	3.27	21	0.357
24	13:58:49	3.42	22	0.374
25	13:58:49	3.59	23	0.39
26	13:58:49	3.75	24	0.406
27	13:58:49	3.91	25	0.422
28	13:58:49	4.06	26	0.437
29	13:58:49	4.24	27	0.453
30	13:58:50	4.39	28	0.469
31	13:58:50	4.55	29	0.484

Move this sheet to 1st tab position to accept data

Port: 4

Baud: 9600

[✓] Reset on Connect

Restarting Data

Accepting data for Row 281

BASIC Stamp Code
BASIC Stamp code is used to illustrate using the Simple Data with Plotting sheet of PLX-DAQ.

- This program sends the value of variable X and the BASIC Stamp SIN value of X.

SEROUT sPin,Baud,["DATA,TIME,TIMER,", DEC X, ",", SDEC SIN X, CR]

- The current row is read, and when exceed, will reset the row back to 2.
  SEROUT sPin,Baud,["ROW,GET",CR] ' Request Row
  SERIN sPin, Baud,200,TimeOut,[DEC Row] ' Accept returning data with timeout
  IF Row >= 300 THEN SEROUT sPin,Baud,["ROW,SET,2",CR] ' If Row is or exceeds 300, set row back to 2
- Note that **SEROUT 16,84** (sPin and Baud values) is used to send data. This is equivalent to **DEBUG** but prevent the DEBUG window opening.
- **SERIN 16,84** is used, which is equivalent to **DEBUGIN**, but provides for timeouts to be used so that the program does not lock up if there is a problem.
- When using the DATA directive to send multiple values, a combination of strings and values is needed to force commas between values.

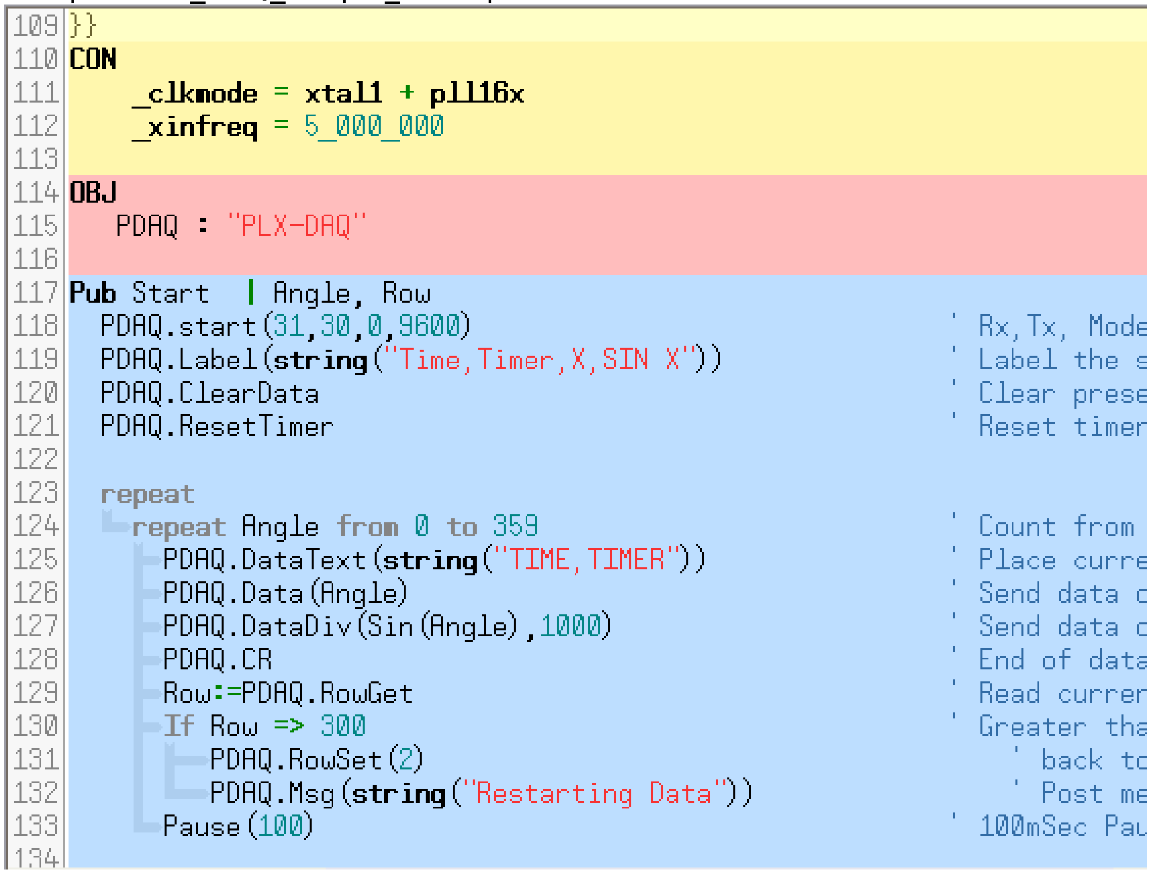
SX/B Code
SX/B Code is used in a similar fashion to the BASIC Stamp code, but since SX/B does not directly support SIN, and the sending of strings and decimal values, routines were included to simulate a wave and to support transmission/reception of strings and decimal values.

```
' Example code to send a string
SendStr "DATA,TIME,TIMER,"

' Example routines to send string data
SendChar: ' Sends a single character
char = __PARAM1
SEROUT SOutPin, Baud, char
RETURN

SendStr: ' Sends a string
tempW = __WPARAM12
DO
READINC tempW, char
IF char = 0 THEN EXIT
SendChar char
LOOP
RETURN
```

Propeller Spin Example Code

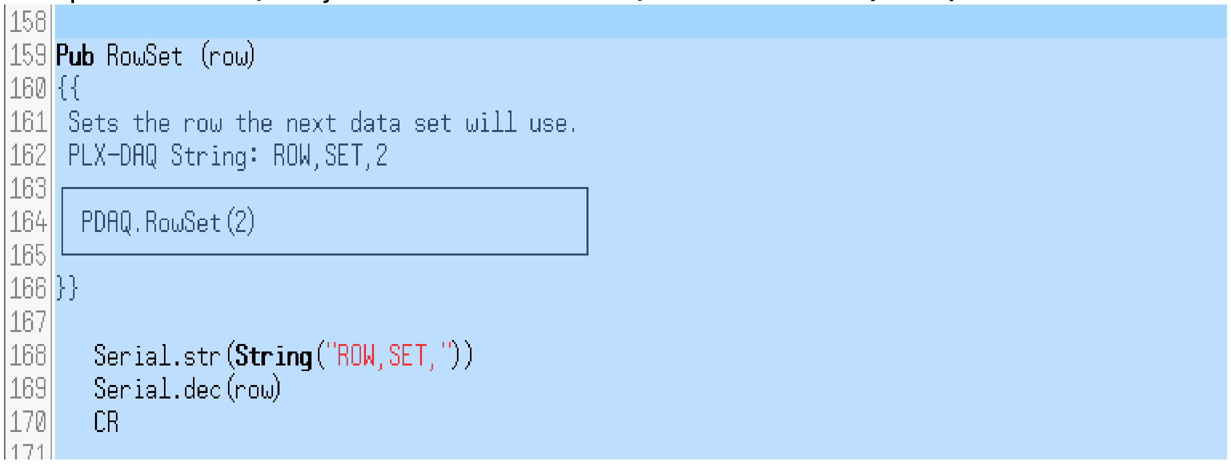
An object has been developed to allow quick use of PLX-DAQ. The **PLX-DAQ object** file requires the **Extended_FDSerial** and **FullDuplexSerial** objects which are included with the example download. The object provides the formatting required by PLX-DAQ for communications.

Sample "PLX_DAQ_Simple_Test.Spin" Code:



```
}}
CON
    _clkmode = xtal1 + pll16x
    _xinfreq = 5_000_000

OBJ
   PDAQ : "PLX-DAQ"

Pub Start   | Angle, Row
  PDAQ.start(31,30,0,9600)                                ' Rx,Tx, Mode
  PDAQ.Label(string("Time,Timer,X,SIN X"))                ' Label the s
  PDAQ.ClearData                                          ' Clear prese
  PDAQ.ResetTimer                                         ' Reset timer

  repeat
    repeat Angle from 0 to 359                            ' Count from
      PDAQ.DataText(string("TIME,TIMER"))                 ' Place curre
      PDAQ.Data(Angle)                                    ' Send data c
      PDAQ.DataDiv(Sin(Angle),1000)                       ' Send data c
      PDAQ.CR                                             ' End of data
      Row:=PDAQ.RowGet                                    ' Read curren
      If Row => 300                                       ' Greater tha
         PDAQ.RowSet(2)                                       ' back to
         PDAQ.Msg(string("Restarting Data"))                  ' Post me
      Pause(100)                                          ' 100mSec Pau
```

Sample "PLX-DAQ" object code for PLX-DAQ directive **ROW, SET,value**



```

Pub RowSet (row)
{{
 Sets the row the next data set will use.
 PLX-DAQ String: ROW,SET,2

  PDAQ.RowSet(2)

}}

   Serial.str(String("ROW,SET,"))
   Serial.dec(row)
   CR

```

The use of the PLX-DAQ object file is very well documented. Please refer to that file for more information.

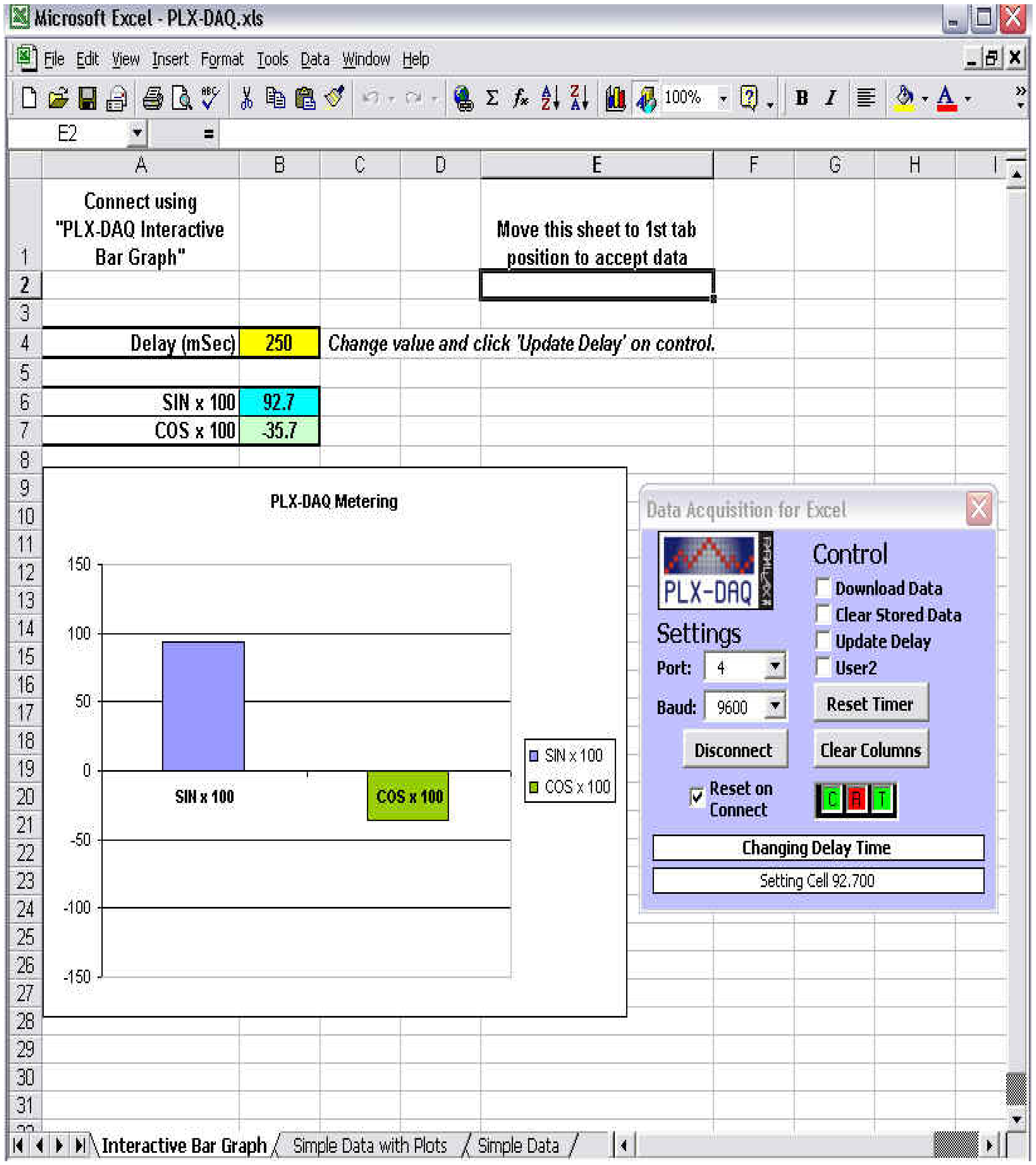
Interactive Metering Example

Interactive Metering/Graphing

This example uses the "Interactive Bar Graph" source code and the worksheet. Example code for the BS2, SX, and Propeller are provided in a separate download available on Parallax's PLX-DAQ page.

The example performs the following:

- Labels cells on the spreadsheet using **CELL,SET**
- Updates the values for SIN and COS and plots them in a bar graph using **CELL,SET**.
- Updates the control's User1 checkbox label to "Update Delay" using **USER1,LABEL,Update Delay**
- Checks the value of the User1 checkbox using **USER1,GET**
- If checked (1), reads the delay (mSec) value using **CELL,GET,B4**



	A	B	C	D	E
1	Connect using "PLX-DAQ Interactive Bar Graph"				Move this sheet to 1st tab position to accept data
4	Delay (mSec)	250	*Change value and click 'Update Delay' on control.*		
6	SIN x 100	92.7			
7	COS x 100	-35.7			

Legend: ▨ SIN x 100, ▨ COS x 100

Control: ☐ Download Data, ☐ Clear Stored Data, ☐ Update Delay, ☐ User2

Port: 4

Baud: 9600

☑ Reset on Connect

Changing Delay Time

Setting Cell 92.700

BASIC Stamp Code Examples

BASIC Stamp code is used to illustrate using the Simple Data with Plotting sheet of PLX-DAQ.

- This program sends the value BASIC Stamp SIN and COS of variable X to be graphed for virtual metering.
  The delay time of the loop may be read from a cell when a labeled check box is checked.
- The cell value for CELL,SET and CELL,GET uses hexadecimal value for columns A-F.
  ```
  Delay_Cell_Label CON $A4
  SEROUT sPin,Baud,["CELL,SET,", HEX Delay_Cell_Label,", Delay (mSec):", CR]
  ```
- Note that **SEROUT 16,84** is used to send data. This is equivalent to **DEBUG** but prevents the DEBUG window opening.
- **SERIN 16,84** is used, which is equivalent to **DEBUGIN**, but provides for timeouts to be used so that the program does not lock up if there is a problem.
- The value of Delay cell is read when the USER1 check box is set. The checkbox is then cleared.
  ```
  SEROUT sPin,Baud,["USER1,GET",CR]
  SERIN sPin,Baud,50,Timeout,[DEC chkDelay]
  ' IF checked, read new delay time
  IF chkDelay = 1 THEN
  SEROUT sPin,Baud,["CELL,GET,", HEX Delay_Cell,CR ]
  SERIN sPin,Baud,50,Timeout,[DEC Delay]
  ' Uncheck USER1 box
  SEROUT sPin,Baud,["USER1,SET,0",CR]
  ENDIF
  ```

SX/B Code
SX/B Code is used in a similar fashion to the BASIC Stamp code, but since SX/B does not directly support SIN, and the sending/reception of strings and decimal values, routines were included to simulate a wave and to support transmission/reception of strings and decimal values.

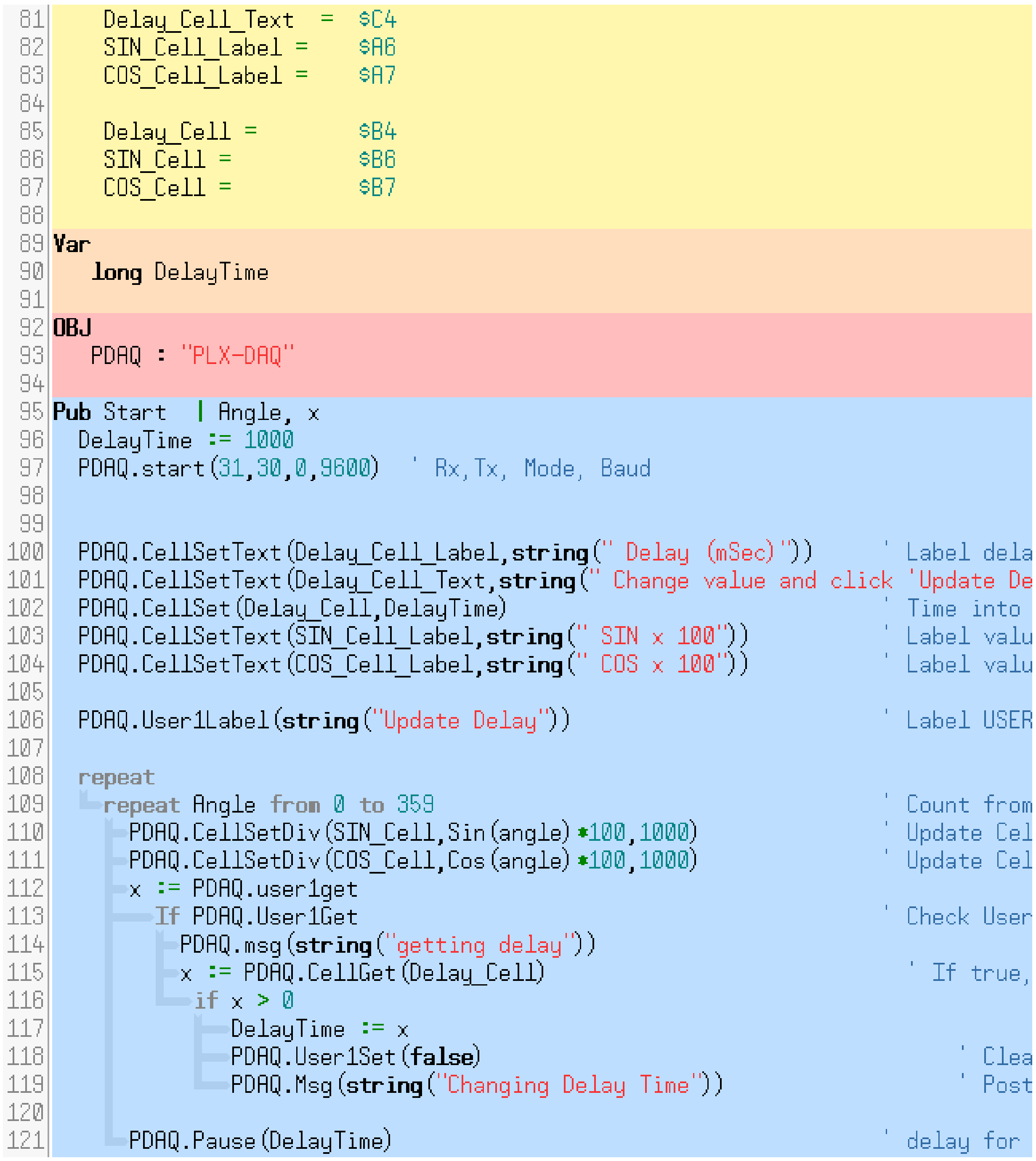
Sample Code:

```
SendStrCR "USER1,GET"
tempW = RecvWord
' If checked, read new delay time
IF tempW = 1 THEN
SendStrCR "CELL,GET,B4"
delay = RecvWord
' Uncheck USER1 box
SendStrCR "USER1,SET,0"
ENDIF
PAUSE delay
```

Propeller Spin Example Code

An object has been developed to allow quick use of PLX-DAQ. The **PLX-DAQ object** file requires the **Extended_FDSerial** and **FullDuplexSerial** objects which are included with the example download. The object provides the formatting required by PLX-DAQ for communications.

Sample "PLX_DAQ_Interactive_Bar_Graph.Spin" Code:



```
    Delay_Cell_Text  =  $C4
    SIN_Cell_Label =    $A6
    COS_Cell_Label =    $A7

    Delay_Cell =        $B4
    SIN_Cell =          $B6
    COS_Cell =          $B7

Var
   long DelayTime

OBJ
   PDAQ : "PLX-DAQ"

Pub Start  | Angle, x
  DelayTime := 1000
  PDAQ.start(31,30,0,9600)  ' Rx,Tx, Mode, Baud


  PDAQ.CellSetText(Delay_Cell_Label,string(" Delay (mSec)"))      ' Label dela
  PDAQ.CellSetText(Delay_Cell_Text,string(" Change value and click 'Update De
  PDAQ.CellSet(Delay_Cell,DelayTime)                               ' Time into
  PDAQ.CellSetText(SIN_Cell_Label,string(" SIN x 100"))            ' Label valu
  PDAQ.CellSetText(COS_Cell_Label,string(" COS x 100"))            ' Label valu

  PDAQ.User1Label(string("Update Delay"))                          ' Label USER

  repeat
    repeat Angle from 0 to 359                                     ' Count from
      PDAQ.CellSetDiv(SIN_Cell,Sin(angle)*100,1000)                ' Update Cel
      PDAQ.CellSetDiv(COS_Cell,Cos(angle)*100,1000)                ' Update Cel
      x := PDAQ.user1get
        If PDAQ.User1Get                                           ' Check User
          PDAQ.msg(string("getting delay"))
          x := PDAQ.CellGet(Delay_Cell)                              ' If true,
           if x > 0
              DelayTime := x
              PDAQ.User1Set(false)                                     ' Clea
              PDAQ.Msg(string("Changing Delay Time"))                  ' Post

      PDAQ.Pause(DelayTime)                                        ' delay for
```