

Basic Micro Application Note : Oscilloscope

Contents

- Introduction
- Overview
- Oscilloscope controls
- Sample code

Introduction

There is an oscilloscope tool included in the Basic Micro IDE. For those who do not have an oscilloscope, which can be very expensive, this tool can really come in handy and at the price is right. Free.

You must keep in mind this is not a replacement for a standalone oscilloscope, nor is it intended to be. No software based oscilloscope can be due to the nature of most PC operating systems. They are just not meant to be true real time systems and if nothing else that's the true purpose of an oscilloscope, real time information.

Overview

As stated above, this is a software oscilloscope. This means that data is sent from a micro controller to a PC. The software (oscilloscope) then reads the data, does what ever calculations that need to be done and plots the data.

The time base can only be as fast as the micro controller sending the data and the speed of the PC receiving the data. I feel it is very important to understand this. Do not expect to get the same results you'd get from a standalone oscilloscope.

That said, this is a very useful tool and will generally handle most jobs a hobbyist is likely to use it for.

The oscilloscope supports two basic kinds of data plots. Y/time where the Y axis is the data and the X axis is time. Note that this is not real time. The time is actually the time between when a data point is received. So if your only sending a data point once a second that's when the Y axis will be updated.

Here we'd send four bytes for the data, a long for a Y/time plot:

```
serout B0,i9600,[0x0e,Result.byte0,Result.byte1,Result.byte2,Result.byte3]
```

The other kind of data plot is Y/X where X and Y are two separate data sets in relation to the speed of the screen update (the scrolling of the X axis). This option is only available for channel 17(0x1F)

Here we'd send four bytes for the data, but this time as two words :

```
serout B0,i9600,[0x0e,Result0.byte0,Result0.byte1,Result1.byte0,Result1.byte1]
```

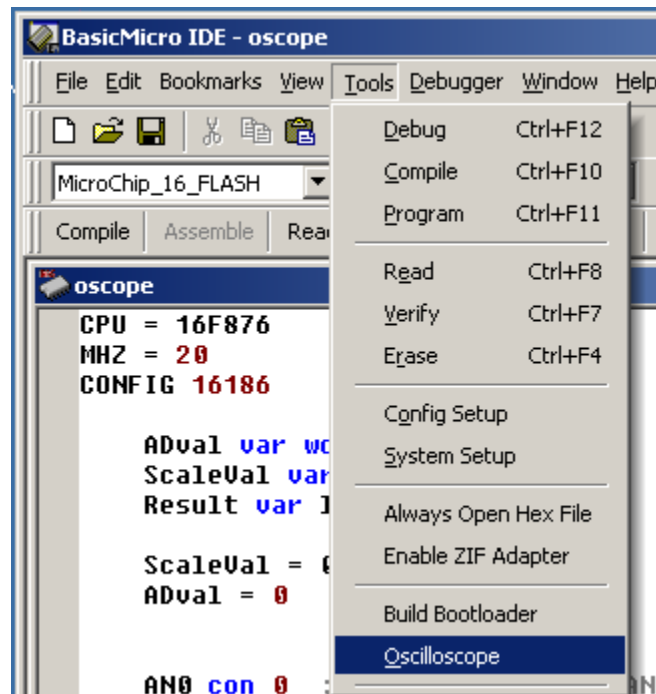
Another great feature is that up to 16 Y/time channels can be plotted. 0x0e – 0x1e where 0x0e (14) is channel 1 and 0x1ef (30) is channel 16.

Important note.

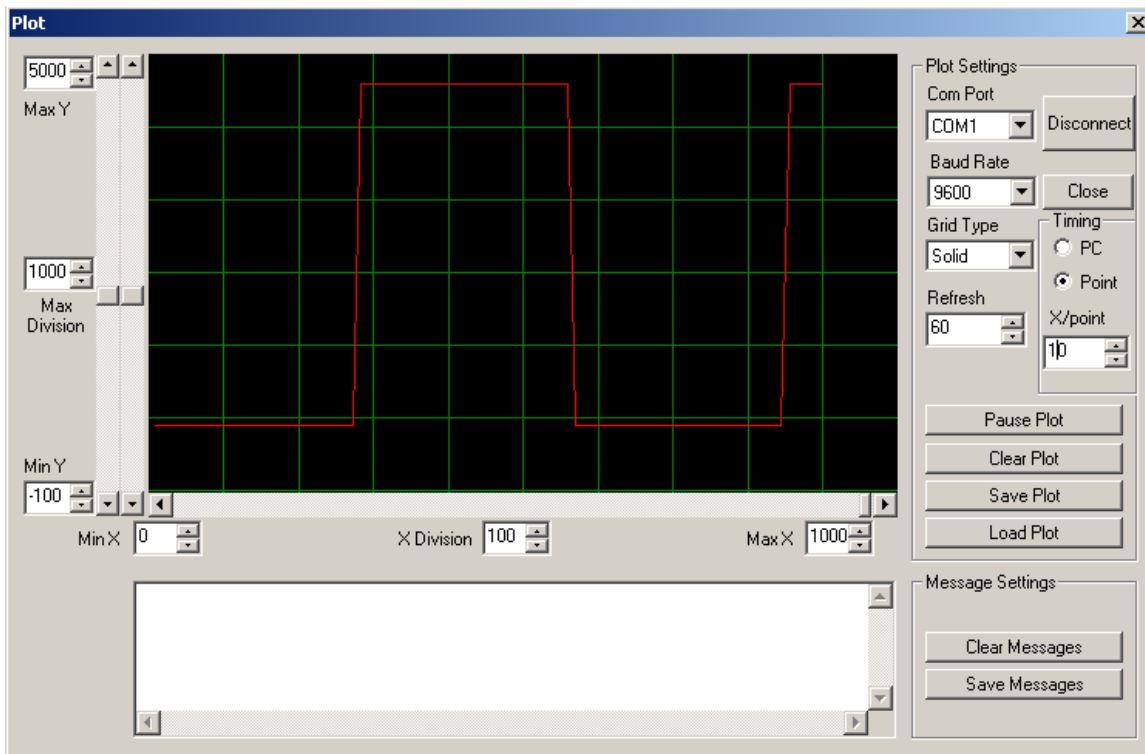
The oscilloscope only works with integer values. A floating point value can not be sent to the oscilloscope. As an example if the value is 3.583 then that would have to be multiplied by 1000 to send a value of 3583 and the Y axis scale would have to be set to 5000 max Y and 0 min Y if 0-5 volts was being plotted.

Oscilloscope controls

The oscilloscope is found in the tools menu of the Basic Micro IDE



When the oscilloscope is selected the following screen will be presented.



*** Note : The following control descriptions are subject to change ***

We'll start top to bottom and left to right.

Max Y : This is the max Y scale setting in integer units.

Max division : This is the number of integer units per Y grid line.

Min Y : This is the min Y scale setting in integer units.

Min X : This is the min X scale setting in integer units.

Max division : This is the number of integer units per X grid line.

Max X : This is the max Y scale setting in integer units.

Port Setting : Selects the comm. Port and baud rate. If the baud rate is over 57.6 then some data loss is possible depending on your PC speed.

Message settings : This is used for debugging



Grid type: Changes the grid display

Refresh : How often the screen is re painted or refreshed. This is displayed in Hz

Pause plot : Pauses the plot

Clear plot : Clears the plot

Save plot : Saves the current plot to a file.

Load plot : Loads a saved plot

Connect / Disconnect: Starts or stops the oscilloscope comms.

Close : Ends the program.

Timing : The PC option will increment the X axis every N – us. Where N is the number of micro seconds entered. Point is one X axis point per data point over time.

Sample code 1

This sample simply toggles a port from high to low and sends the data to the oscilloscope. Connect port b1 to port ao (ADC 0).

```
CPU = 16F876  
MHZ = 20  
CONFIG 16186
```

```
ADval var word  
ScaleVal var long  
Result var long  
I var byte  
ScaleVal = 0.0048 'this is used to scale the AD reading to 0-5 volts
```

ADval = 0

AN0 con 0 ;this sets the pin AN0 / RA0,
CLK con 2
ADSETUP con %10001110

output b1

main

high b1
gosub rdad
pause 100
low b1
gosub rdad
pause 100
goto main

rdad

for I = 0 to 25 'we do this a number of times to get a nice square wave.

ADIN AN0,CLK,ADSETUP,ADval ;ADval holds an integer

Result = (float ADval) fmul ScaleVal '*****should be 0-5v in .0048 per bit *****

Result = Result fmul 1000.0 ;multiply voltage by 1000 to get 3 decimal places of precision sent
back to o-scope ;O-Scope should be setup for a 0 to 5000 range.

Result = int Result ;convert to a long integer so serout can send the value to o-scope correctly

serout B0,i9600,[0x0e,Result.byte0,Result.byte1,Result.byte2,Result.byte3]

next

return

Sample code 2

This sample reads the voltage from a 10K pot connected between VDD, VSS and the ADC and sends the data to the oscilloscope.

```
CPU = 16F876
MHZ = 20
CONFIG 16186
```

```
ADval var word
ScaleVal var long
Result var long
ScaleVal = 0.0048
ADval = 0
```

```
AN0 con 0 ;this sets the pin AN0 / RA0,
CLK con 2
ADSETUP con %10001110
```

```
main

    gosub rdad

goto main

rdad

    ADIN AN0,CLK,ADSETUP,ADval ;ADval holds an integer

    Result = (float ADval) fmul ScaleVal '*****should be 0-5v in .0048 per bit *****

    Result = Result fmul 1000.0 ;multiply voltage by 1000 to get 3 decimal places of precision sent
    'back to o-scope

;O-Scope should be setup for a 0 to 5000 range.

    Result = int Result ;convert to a long integer so serout can send the value to o-scope correctly

    serout B0,i9600,[0x0e,Result.byte0,Result.byte1,Result.byte2,Result.byte3]

return
```

Sample code 3

This sample generates a sin wave that “rolls back” on itself and sends the data to the oscilloscope. This is an example of an Y/X vice a Y/ time plot.

```
CPU = 16F876
MHZ = 20
CONFIG 16186
```

```
cnt var byte
cnt2 var byte
temp var word
temp2 var word
cnt = 0
cnt2 = 255
```

```
main
```

```
temp = (sin cnt)+128
temp2 =(cos cnt2)+384
```

```
cnt = cnt + 1
cnt2 = cnt2 -1
'serout B0,i9600,[dec temp2,dec temp,13] 'send to message window
serout B0,i9600,[0x1f,temp2.byte0,temp2.byte1,temp.byte0,temp.byte1]
goto main
```

Atom Sample code 1

This sample simply toggles a port from high to low and sends the data to the oscilloscope.
Pin p0 is connect to AX0 (ADC 0)

```
ADval var word
ScaleVal var long
Result var long
I var byte
ScaleVal = 0.0048 'this is used to scale the AD reading to 0-5 volts
ADval = 0
```

```
AN0 con 0 ;this sets the pin AN0 / RA0,
CLK con 2
ADSETUP con %10001110
```

```
output p0
```

```
main
```

```
high p0
gosub rdad
pause 100
low p0
gosub rdad
pause p0
goto main
```

```
rdad
```

```
for I = 0 to 25 'we do this a number of times to get a nice square wave.
```

```
ADIN AN0,CLK,ADSETUP,ADval ;ADval holds an integer
```

```
Result = (float ADval) fmul ScaleVal '*****should be 0-5v in .0048 per bit *****
```

```
Result = Result fmul 1000.0 ;multiply voltage by 1000 to get 3 decimal places of precision sent
back to o-scope ;O-Scope should be setup for a 0 to 5000 range.
```

```
Result = int Result ;convert to a long integer so serout can send the value to o-scope correctly
```

```
serout S_out ,i9600,[0x0e,Result.byte0,Result.byte1,Result.byte2,Result.byte3]
```

```
next
```

```
return
```

Atom Sample 2

This sample reads the voltage from a 10K pot connected between VDD, VSS and the ADC and sends the data to the oscilloscope.

```
ADval var word
ScaleVal var long
Result var long
ScaleVal = 0.0048
ADval = 0
```

```
AN0 con 0 ;this sets the pin AN0 / RA0,
CLK con 2
ADSETUP con %10001110
```

```
main
```

```
    gosub rdad
```

```
goto main
```

```
rdad
```

```
ADIN AN0,CLK,ADSETUP,ADval ;ADval holds an integer
```

```
Result = (float ADval) fmul ScaleVal '*****should be 0-5v in .0048 per bit *****
```

```
Result = Result fmul 1000.0 ;multiply voltage by 1000 to get 3 decimal places of precision sent  
back to o-scope
```

```
;O-Scope should be setup for a 0 to 5000 range.
```

```
Result = int Result ;convert to a long integer so serout can send the value to o-scope correctly
```

```
serout s_out,i9600,[0x0e,Result.byte0,Result.byte1,Result.byte2,Result.byte3]
```

```
return
```