Simulator for Arduino

Table of contents

Introduction 4
- What's new 4
- Free Version Differences 5
- Simulator or Arduino 5
- Help 6

Getting Started 6
- System requirements 6
- Getting help 6
- Unlocking the Simulator 7

Using the IDE 7
- Program Window 8
  - Legend 9
  - Program Popup Menu 9
  - BreakPoint 10
  - Loading a Sketch 11
  - Running a Sketch 11
  - Hidden code 11
  - Stimulus File 12
  - <Graphics.h> virtual Library 12
- Variables Area 13
  - Variables Popup Menu 14
- Arduino Area 14
  - Arduino Popup Menu 15
- Menu System 15
  - File Menu 16
  - Run Menu 16
  - View Menu 17
  - Hardware 17
  - Tools 19
    - Logic Analyzer 19
    - Serial Monitor 20
    - Unit Testing 21
  - Trace 22
  - Options 23
  - Help 23
- Dialog Windows 24
  - Edit Window 24
    - Bare Minimum 25
  - EEPROM Window 26
  - Input/Output Window 27
    - Simulation Popup Menu 28
  - View Subroutines 29
  - Call Stack 29

- Class Explorer 30
- Error Message 30
- Shortcut Toolbar 30
- Shortcut keys 31
- Example Sketches 32
- Test Sketches 32
- Language Processing 32
- Keywords 32
- Custom Libraries 33
- Millis 33
- Troubleshooting 33
- Registry Settings 34
- Version Info 34
- Credits 36

Introduction

The Arduino development system is a powerful tool but has no debugging or emulation capabilities. The Simulator for Arduino is the best debugging tool available and is ranked #1 on most review sites.

Simulator for Arduino is an Arduino Simulator which may be downloaded or purchased from www.virtronics.com.au. The Free version has a 30 minute trial, a thirty day trial period (or 300 seconds until unlocked) or 100 sketches then a 30 second incrementing delay on opening sketches and is code limited to 200 lines. The Pro Version has no limitations, and is supplied with updates until the end of the calendar year. A subscription option with >50% discount is then available to renew the Pro licence for another year.

The benefits and features of an Arduino Simulator are:

- Use Arduino without needing the hardware
- Teach, educate and investigate the basics of Arduino sketches by stepping through code
- Debug a sketch, step through or run with a breakpoint and see the variables change
- Speed up Arduino designs and trial other Arduino boards first and see the pin-out configurations
- Demonstrate a project to a potential customer remotely
- Discover number overruns such as setting a byte to 256, assigning a new value to a const variable, or use library routines without including the library
- Graphically view LCD and colour screens with the inbuilt Graphics Library
- Trace and Error logs
- Many Tools such as an Improved Serial Monitor, a Simple logic analyser to view digital pins graphically and Automated self testing (Unit Testing)
- Variable insight which allows for variables values to be viewed by clicking on them in the Sketch Window (ensure Menu Option is turned on)
- Design a new board inside the Simulator using the Hardware > Save Settings and Load Settings items.
- Many real-world user sketches to provide inspiration found in the _Sim_Test folder
- Many more features and continuous improvements with regular updates planned

The SIMFORARDUINO Pro Version license is licenced to one user only for use on only two computers, with a condition that any other user can use the Simulator for up to one hour for educational use only. SiteWide licences with generous discounts are available to Educational institutions for 25 or more seats. Email Virtronics for more info (support@virtronics.com.au)

Press F1 inside the Simulator to open context sensitive help.

What's new

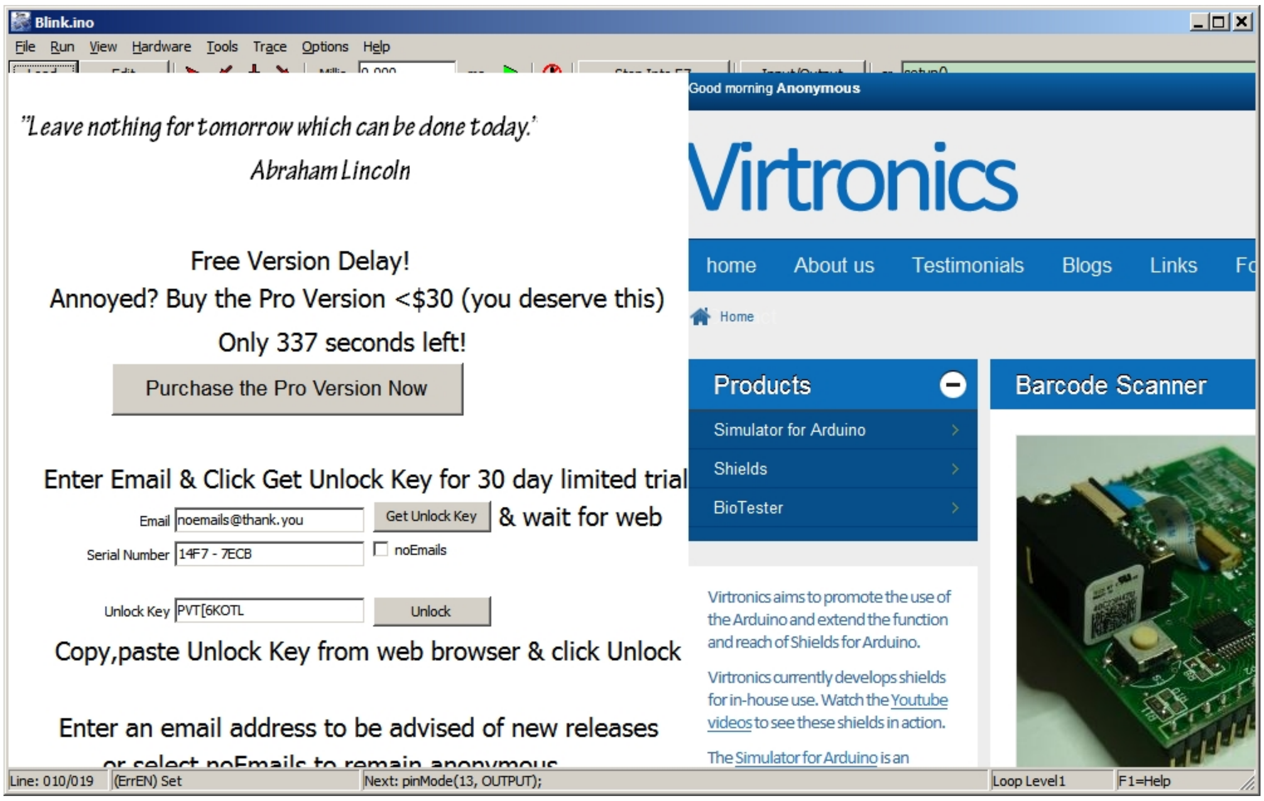
Version 1.10 Jan 2018
- fix micros
- and/or priority
- fix strcpy
- add ^= operator
- improve define includes - allow for #define(a,...)
- add new directive SIMULATING - true when runnign Simulator - #ifdef SIMULATING
- Click on Variable Valu etext to change radix - binary => Char => Decimal => Hex => Binary

Free Version Differences

The Free Version is identical to the Pro Version, apart from the addition of a timer delay window and a Code Limit of 200 lines. The Free version is supplied as a demo version to allow users to trial the Simulator for Arduino program. The logic and operation of the Simulator for Arduino program is identical apart from the limitations. The Pro Version License comes with the latest Simulator version and Upgrades until the end of the Calendar year, and has been purchased by over 7000 users and many educational institutions. The Free version users may receve more emails.

Click the Get Unlock key to receive a free 30 day trial with no timer but with the 200 line limit and 100 sketches. After Unlocking, the Simulator will appear with the time and sketch loads left on the Status bar as shown here. After 30 days the timer will be reset to 30 seconds for each new sketch loaded or edited.

Enter a valid email address to receive emails about updates, or click the noemails checkbutton otehrwise.



"Leave nothing for tomorrow which can be done today."
Abraham Lincoln

Free Version Delay!
Annoyed? Buy the Pro Version <$30 (you deserve this)
Only 337 seconds left!

Enter Email & Click Get Unlock Key for 30 day limited trial

Email noemails@thank.you & wait for web
Serial Number 14F7 - 7ECB

Unlock Key PVT[6KOTL
Copy,paste Unlock Key from web browser & click Unlock

Enter an email address to be advised of new releases

Virtronics aims to promote the use of the Arduino and extend the function and reach of Shields for Arduino.

Virtronics currently develops shields for in-house use. Watch the Youtube videos to see these shields in action.

The Simulator for Arduino is an

Line: 010/019 | (ErrEN) Set | Next: pinMode(13, OUTPUT); | Loop Level 1 | F1=Help

Simulator or Arduino

Simulator or Arduino is a good question and *Simulator and Arduino* or even *Arduino and then Simulator* is a good answer. The Simulator was not designed as a replacement for Arduino but an add-on product to the Arduino development kit and has been designed as a training/demo/debugging tool. It has not been designed as a development or compiling tool. We fully support the Arduino project and encourage new users to buy and try a real Arduino kit first and then purchase the Simulator as needed. For first time users, please refer to the book "Getting Started with Arduino" by Massimo Banzi.

To develop code for Arduino, a good starting point is to use the Arduino Uno together with the Arduino IDE. For larger projects, the Arduino Mega or Due may be more suitable. This Simulator has not been designed to check for syntax errors, and we recommend only using the Simulator for debugging only after the Arduino sketch compiles in the Arduino IDE but there is still some issue preventing correct operation.

Official Arduino kits are recommended since these are well designed and support for the Arduino project leads to more great new development boards and IDE upgrades. Please note that the official arduino website is www.arduino.cc and we recommend following Massimo Banzi and David Cuartailles on Twitter for the latest Arduino news.

Help

The Simulator for Arduino is designed to be as easy to use as possible and bug-free. However, this depends on the level of complexity of the Arduino sketch being simulated and from time to time, the Simulator may not work as expected. In these rare and very infrequenct cases, please email support@virtronics.com.au or else post on the forum at forum.virtronics.com.au and someone at Virtronics will try to reply as soon as possible - usually withni a few days.

Getting Started

Getting Started.

Download the zip file, extract the setup.exe file. When finished, run this program to install the Simulator for Arduino program and click yes to accept the EULA (End User Licence Agreement). For the Free Version, there is a 30 minute trial period then an unlock process. To Unlock the Simulator, enter the email address, the serial number will be auotmatically read fromthe computer, and then click on the Get Unlock Key button. Copy the Unlock Key back to the Simulator and clicking Unlock. Please email us if there are any issues with this process.

Afetr purchasing a Pro Version Licence, an email wil be sent with instructions on how to download the separate Pro Version. This is done by a third party partner SWPAL.com and the email should be sent out soon. If the email is not received within a reasonable time, please check in Junk or SPAM mail. Otherwise, email us at support@virtronics.com.au and we will try to reply as soon as possible with download and installation instructions.

System requirements

The System Requirements are a PC Pentium II or later.

The Simulator for Arduino program will run on the following systems:

WINDOWS

- Windows 2000
- Windows ME
- Windows XP
- Windows Vista
- Windows 7
- Windows 8 and 8.1
- Windows 10

MAC

- Windows using Parallels
- Windows using VMWare Fusion

LINUX

- Wine on Linux (windows emulator) for version 16.04

Getting help

For help, please email support@virtronics.com.au and please attach the sketch, or click the email button inside any Error Message. Replies will be sent as soon as possible.

Also, please make use of the forum (http://forum.virtronics.com.au/) . Subscribe to the thread at the lower left of the webpage to be emailed about any replies. There are already many questions and answers on the forum and threads can be subscribed to in order to be emailed about new replies. Click the little box in the lower part of the screen to subscribe to a thread.

Before emailing, please check that the sketch compiles in the Arduino IDE first.

Unlocking the Simulator

The Unlock process was originally added to stop pirates from cracking the Simulator. It allows a list of user's email addresses to be gathered and then an email is sent out to this mailing list for each update which usually happens at the end of each quarter. If an email update is not required, click the noemails button or enter a random email.

With the Unlock, the email address and serial number are required. If the serial number is empty, please restart the Simulator. The serial number is read from the computer hard drive and should be automatically read by the Simulator on startup. Another option is to select Help > Unlock key. When these fields are filled, click the Get Unlock button. A webpage will then open with the Unlock key - this may take a few seconds for the web browser to open and load the webpage. Copy and paste this back into the Simulator and press Unlock. The same process is required for the Free and Pro versions. Please watch the video below for a real-wrold Unlocking of the Free version
https://youtu.be/z3tUEiG3Ts4?list=PLvqNlsdvRBNTDAZkUYAwc3cWYRo5wPvLK

Using the IDE

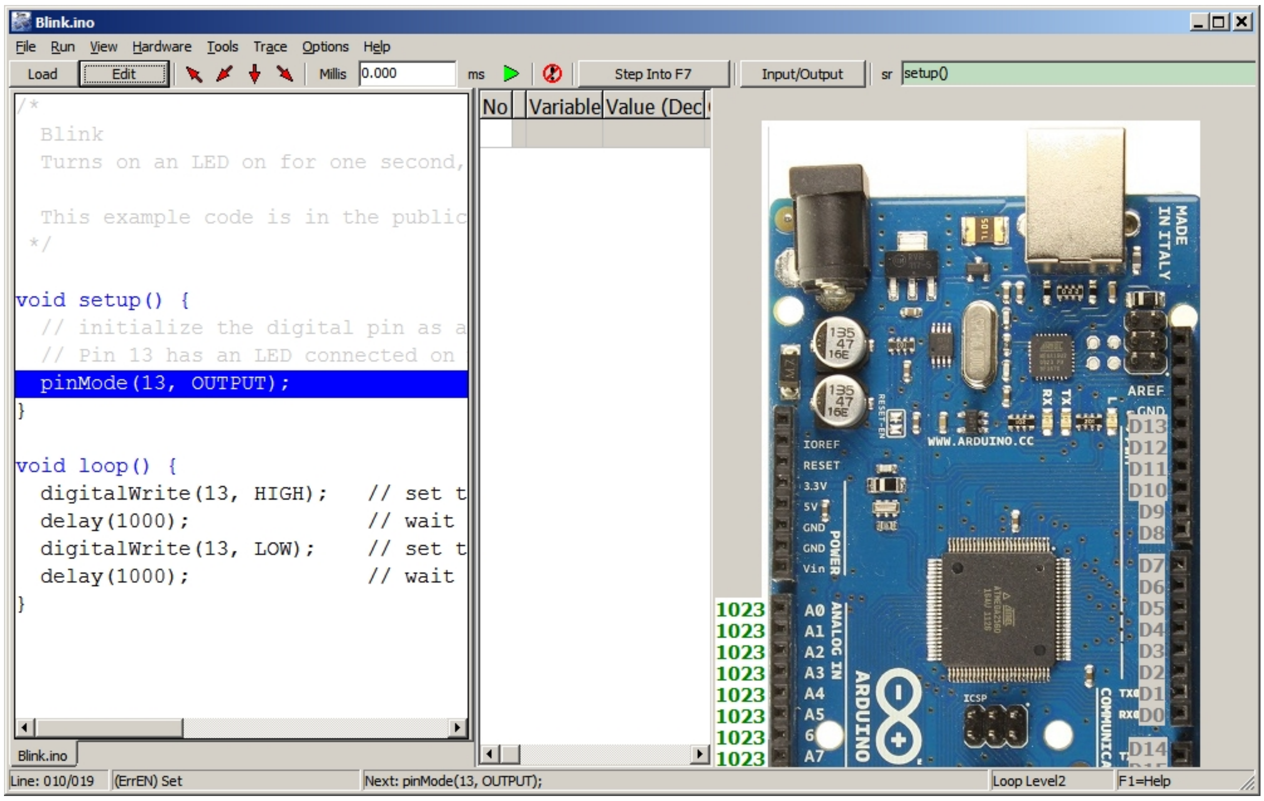
The Simulator for Arduino IDE has several sections. These are:

- The Menu system which contains options and settings
- The Shortcut Toolbar provides single click action for commonly used operations
- The Program or Sketch window
- The Variables area
- The Arduino Area

The order of the Program window, Variables are and Arduino picture may be swapped by Selecting View| Arduino on Left. The preferred method is to have the Program Window on the left so that the work flow is input,data and output. The Arduino picture and/or Variables area may be hidden by selecting Options > Show Arduino and/or Show Variables and clicking on it to uncheck these options.

When the Simulator for Arduino IDE is resized, the Variables window will disappear first since this is seen as the least important window. If the main screen is made smaller, the Program window will then disappear and the Arduino Screen will always be on top. This allows for sketches to be simulated as they would appear for real. There is also a Minimize button designed for this purpose.

Note that the Variables Area and the Arduino picture can be undocked by using the Allow Dock/Undock in the Variables Area right click popup menu.



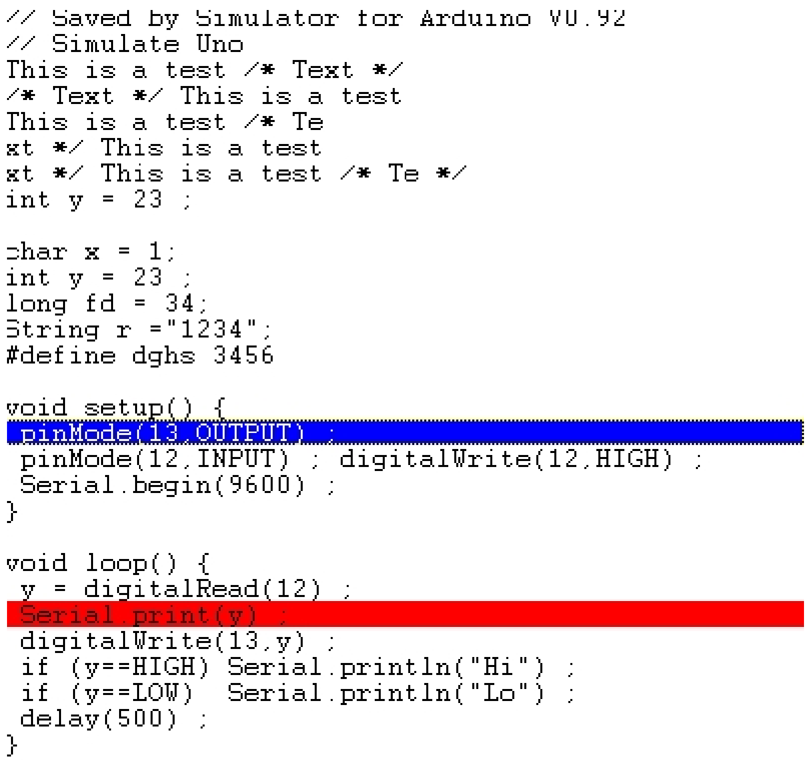
```
/*
  Blink
  Turns on an LED on for one second,

  This example code is in the public
 */

void setup() {
  // initialize the digital pin as a
  // Pin 13 has an LED connected on
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set t
  delay(1000);              // wait
  digitalWrite(13, LOW);    // set t
  delay(1000);              // wait
}
```

```
Line: 010/019   (ErrEN) Set   Next: pinMode(13, OUTPUT);   Loop Level2   F1=Help
```

Program Window

The Program window area shows the program listing, and the cursor will always be in the vertical center of the screen. The Program Window allows for the program to be stepped through. When selected, the following keys may used to run the program:

- F2 may be used to force a reset
- F3 may be used to open the Find window to search for text
- F6 may be used to edit the sketch
- F7 key may be used to step through the program and
- F8 can be used to step over subroutines while still executing all the code inside the subroutine.
- Shift F8 may be used to step out of a routine
- F9 may be used to run the program
- See the Shortcut Toolbar for more info on the step and run icons



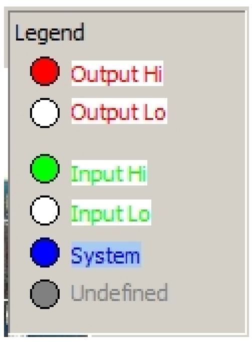
```
// Saved by Simulator for Arduino V0.92
// Simulate Uno
This is a test /* Text */
/* Text */ This is a test
This is a test /* Te
xt */ This is a test
xt */ This is a test /* Te */
int y = 23 ;

char x = 1;
int y = 23 ;
long fd = 34;
String r ="1234";
#define dghs 3456

void setup() {
 pinMode(13,OUTPUT) ;
 pinMode(12,INPUT) ; digitalWrite(12,HIGH) ;
 Serial.begin(9600) ;
}

void loop() {
 y = digitalRead(12) ;
 Serial.print(y) ;
 digitalWrite(13,y) ;
 if (y==HIGH) Serial.println("Hi") ;
 if (y==LOW)  Serial.println("Lo") ;
 delay(500) ;
}
```

A new line may be selected by right clicking anywhere in the Program or Sketch window and selecting Set Next Statement.

The Statusbar shows the Line number, Last Line number and the next code to be executed. If the AutoStep time is set to less than 2ms, the Status-Bar will not update until Run or AutoStep is stopped.

Special Simulator hidden code may be used to setup a particular hardware configuration, and this allows the setup configuration to be saved inside the comments of the sketch. See Hidden code for more detail.

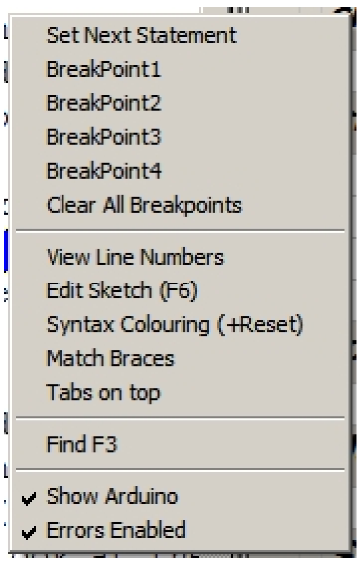
Legend



The Legend may be turned on by selecting View | Legend or right clicking onteh Arduino picture and selecting Legend.

The Legend shows the colours and states of the Digital output. The AnalogWrite value of a digital pin will be displayed to the left of the digital pin when the analogWrite() routine is used.

Program Popup Menu



The Program Area Popup menu has these options:

- Set Next Statement - change the program line
- Breakpoint1-4 - right click to add a breakpoint in red or clear it. Left click on the breakpoint to see which number it is. Also refer to breakpoint condition in the Edit Window.
- Clear All breakpoints - clears all breakpoints with one click

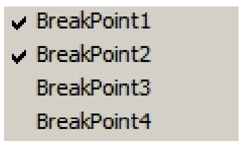
- View Line Numbers - turn on the line numbers in the Program Window - this is useful for checking subroutines
- Edit Sketch - allows the sketch to be edited in the Edit Window
- Syntax Colouring - show comments as light gray, #if hash commands in medium grey, definitions in green, constants in Olive and subroutines as blue
- Match Braces - find the opening or closing bracket for the current line
- Tabs On Top - places the tab with the filename at the top if checked or else the bottom

- Find - open the Find window to find text inside the sketch

- Show Arduino - uncheck to hide the Arduino picture and allow more screen area for your super sketch
- Errors Enabled - turn off to prevent any annoying errors, and hope for the best. Please note this must be manually turned on again to show any sketch errors.

BreakPoint

A breakpoint can be set by right clicking anywhere in the program window.

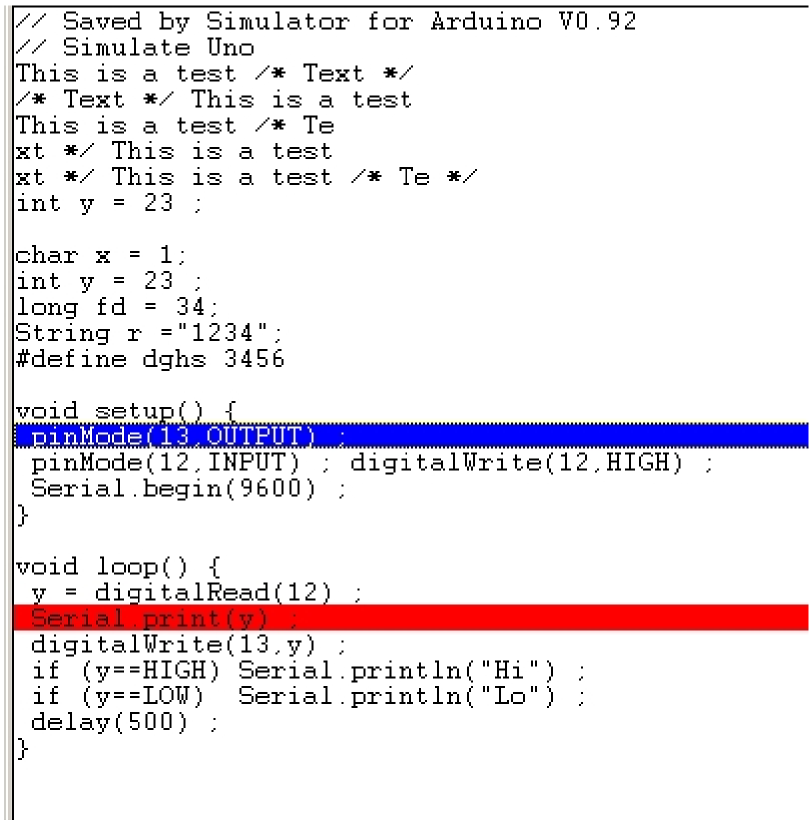


The pop-up menu has an option which can be clicked to add or clear the breakpoint. Please note that a breakpoint condition can be added in the Edit Window to only break at the red line on a certain condition such as when a variable is above a certain level such as i>=17.

If a BreakPoint is added, the breakpoint line will be shown in red. This is commonly used with the AutoStep or F9 command. The breakpoint can also be saved inside a comment using Hidden Code
the syntax

// breakpoint(20)
will set a breakpoint on line 20. If a breakpoint is set on a new line, the previous breakpoint will be cleared. To turn off a breakpoint, select the same line and select Breakpoint.

Left click on a set Breakpoint to find out which number it is. Right click to clear it.



```
// Saved by Simulator for Arduino V0.92
// Simulate Uno
This is a test /* Text */
/* Text */ This is a test
This is a test /* Te
xt */ This is a test
xt */ This is a test /* Te */
int y = 23 ;

char x = 1;
int y = 23 ;
long fd = 34;
String r ="1234";
#define dghs 3456

void setup() {
 pinMode(13,OUTPUT) ;
 pinMode(12,INPUT) ; digitalWrite(12,HIGH) ;
 Serial.begin(9600) ;
}

void loop() {
 y = digitalRead(12) ;
 Serial.print(y) ;
 digitalWrite(13,y) ;
 if (y==HIGH) Serial.println("Hi") ;
 if (y==LOW)  Serial.println("Lo") ;
 delay(500) ;
}
```

Loading a Sketch

To load a sketch, select File|Load Sketch, press the Load Sketch button at the top left or press F4. Navigate to the correct folder and select the *.ino file (or *.pde for previous Arduino versions). C files can now be simulated since the C and Arduino syntax is compatible.

Running a Sketch

After a sketch has been loaded, the first executable line of code will be highlighted. Now press the Step Into button or the down arrow. Alternately, the F7 single step or F8 button may be pressed. The F9 button allows the program to be autorun with a 5ms interval (adjustable -see AutoStep in Edit Window) between steps. The interval can be set to 500ms to demonstrate in slow motion the operation of a sketch, or 1ms to run as fast as possible with minimal refresh.

The middle section of the screen displays the variables grid, and these are organised into Bytes, Chars, Int(eger)s, Long, Strings and Defines. Any value in the grid can be changed by selecting it and typing in a new number.

Hidden code

Hiiden code is code which is hidden inside comments. In this way, the Arduino IDE will skip these comments, but the Simulator will pick up the commands and perform special actions such as setting the Serial input data with a special character string. Hidden code may be used to setup a particular hardware configuration.

This hidden code can be one of these formats:

- // Simulate(Uno) - Simulate the particular Arduino board - please note that Simulate(Uno_LCD.txt) will also work if the text file is found
- // breakpoint(line,tab, condition) - will setup a breakpoint at the line on the correct Tab, (leftmost=1) with the condition. If the condition is left blank, it will not be set
- // SimulateLCD(x,y,width,height,pixel width,pixel space, backlight color, on color, off color) will setup a LCD area on the Arduino picture. The colors are 24bit in a BGR format where 255=Red.The pixel width should be no more than 5)
- // SimulateLED(x,y,width,height,color) where color os the color of LED segments for a 7 segment display
- // SerialIn(data) will load the Serial Port input data with the data inside the parentheses
- // Stimulus(stim.sti) will load a Stimulus file. Stimulus actions are shown in the statusbar after the Next line data and will happen at set times. View the _Sim Test\v98g\test.sti file for a sample. See here for more info.

Stimulus File

The Stimulus file is a text file which allows for digital pins to be set or cleared and predetermined times. Since the Stimulus file accepts any Arduino code, any combination of actions can be set to happoen at specific microsecond runtimes.

The Stimulus file consists of one record per line, with a microsecond time setting, and an expression separated by a comma

Sample Stimulus file

```
20,pinMode(1,OUTPUT); // at 20us make digital Pin1 an output
50,pinMode(5,OUTPUT); // at 50us make digital Pin5 an output
100,digitalWrite(5,HIGH); // at 100us turn on digital Pin5
```

Note that there is also a virtual library <Graphics.h> which allows for colour screen rendering. See the Graphics.ino demo sketch under _Sim Test\v0.98 checks

<Graphics.h> virtual Library

The Graphics library is a virtual library that can simulate real world graphics. For example, in the folder _Sim Test\99f , there is a sketch called UTFT... ino which uses the UTFT or Universal TFT colour LCD display library. SInce it is difficult to simulate a fhigh level LCD controller like the SSD1289 in real-life, the virtual grpahics library makes this much easier. Simply define the Graphics library to have the left,top,width, and height and then the brush and pen colors.

```
//UTFT myGLCD(SSD1289,11,13,10,8,9);

Graphics myGLCD(10,10,320,240,0xffffff,0xffffff);

     brushColor(col)
     circle(x,y,r) // draw a filled circle
     clear() or clrScr()
     drawCircle(x,y,r) // draw an outline of a circle
     drawPixel(x,y,col)
     drawRect(x,y,w,h)
     drawRoundRect
     fillCircle(x,y,r)// draw a filled circle
     fillRect(x,y,w,h)
     fillRoundRect(x,y,w,h,r)
```

```
fillScr(R,G,B)
InitLCD() // dummy routine (does nothing at this stage)
line or drawLine(x,y,x2,y2)
penWidth(width)
penColor(col)
pixel or drawPixel(x,y,col)
print(x,y,s) // print a line of text
printNumI(x,y,i)
rectangle(x,y,w,h)
roundRect(x,y,w,h,r)
setBackColor(R,G,B)
setColor(R,G,B)
setFont(size)
text(x,y,s) // print a line of text
textSize(size)
//textWidth
textColor(col) // set the pen colour to col
```

Variables Area

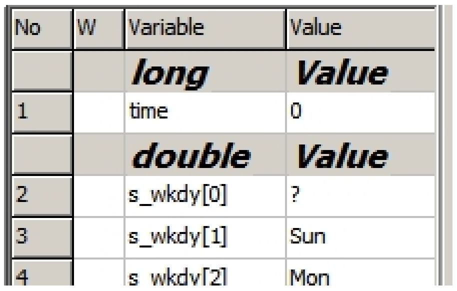
The Variables area shows the status of all the variables used in program and their current value. The Value may be changed by clicking in the relevant cell and changing the value. Please note that having a lot of variables shown can slow the program execution. Either select Watch or Select Options> Show Variables and turn this off to hide the variables to make program execution faster.

Double click in the first four columns to auto-resize the variable area.

Rigth Click and select Binary, Dec, Hex, or Char to change the way teh Value is displayed. For Char mode, arrays can be minimized by pclicking in the W column and the value shown will be string for the char array.

Double Click in the Qualifier column or right click and select Watch to turn on Watch mode. This allows only selected variables to be watched.

The Qualifier column typically shows if the value is a const, unsigned, static, volatile or pointer type (Note that Pointers are not fully supported). The qualifier number in a heading row (shown by the gray shading) shows how many variables of that type have been defined.



No	W	Variable	Value
		long	***Value***
1		time	0
		double	***Value***
2		s_wkdy[0]	?
3		s_wkdy[1]	Sun
4		s_wkdy[2]	Mon

When returning from a subroutine, the stack will be returned the the original state, and any variables defined in the subroutine or passed as arguments will be cleaned or removed.

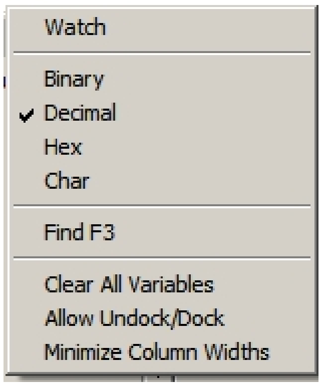
Right Click in the Variables area to bring up the Pop-up Menu.

The second column is the watch column, and clicking in this column will show a w. The right click Pop-up menu can then be selected to show only Watched variables. This is useful when only a few variables are relevant amongst several hundred. The gray area in the W column may be clicked to show or hide all the variables for that type. The last variable changed will always be shown for the last program step.

Arrays can be folded by clicking in the second or Watch column and a + sign will show that the Array has been folded. When folded, only the first element of each array will be shown and for character arrays, the string value of the whole array can be easily viewed. Alternately, the array may be folded at any element of the array and in this case, the elements above will be shown, but the elements below will be hidden. Click the + again to unfold the array

 Clicking in the Qualifier column will enable or disable the watch, and the W in the second column indicates if the Watch is on (W) or off (blank). The last variable changed will always be shown.

Variables Popup Menu



The Variables Area Popup menu has four options:

- Watch - only display variables which have a w or + in the second column of the grid

- Bin - display Variable value in binary
- Decimal - display Variables value in decimal
- Hex - display Variables value in hexadecimal
- Char - display Variables value in Character format and hexadecimal - also show char arrays as strings when the array is minimized by clicking in the w column

- Find - open the Find window to find text inside the variables area

- Clear All Variables - useful for header files which have a #ifdef FILE_H at the beginning. Clearing the variables, then allows the header file to be stepped through. NOTE: doing this in the middle of stepping through a sketch will cause errors since the Simulator will no longer be able to find or set variables.
- Allow Undock/Dock - select this to undock or redock the Variables area from the main SImulator window. If unchecked, the cvolumns may be resized.
- Minimize Column Widths - reduce Column Widths to sensible values to see all Variable Areas

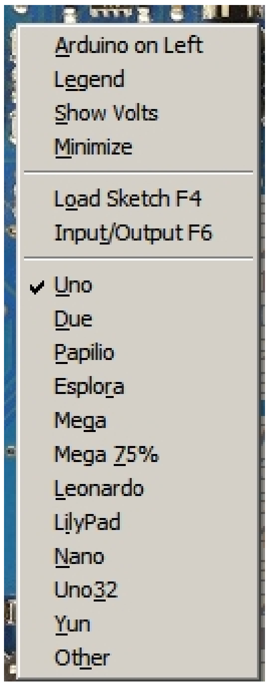
Arduino Area

The Arduino picture area allows for the digital inputs and outputs to be displayed along with the Analog input and output values. The digital pins are defined up to D53 for the Mega and D13 for the Uno.

The Analog Pins are defined as A0 to A5 for the Uno with A0 equal to 14. Other boards are setup according to the Arduino configuration.

The Simulator also allows for custom boards to be setup.

Arduino Popup Menu



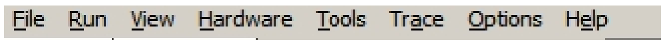
The Arduino Area Popup menu has these options:

- Arduino on Left - display the picture of the Arduino on the left and the Program window on the right
- Legend - display the digital pin legend on the top right of the Arduino picture
- Show Volts - this option allows for the analog number to be displayed as volts in the format x.xx (accurate to 0.01V)
- Minimize - turn off the Tool bar and menu to make the Arduino picture as minimal as possible. the program can be resized to then only show the Arduino picture and the Title (which shows the sketch name).

- Load Sketch - load a sketch into the program window - useful in minimize mode
- Input/Output - show the input/output window - useful in minimize mode

- Uno - set the Arduino board to a Uno type
- Due
- Papilio - great value FPGA platform for under $50
- Esplora
- Mega - set the Arduino board to a Mega type
- Mega 75% - same as above but with a small picture to fit most laptop screens
- Leonardo
- LilyPad - wearable Arduino
- Nano
- Uno32 - Microchip Chipkit - a copy of the Arduino Uno but running a Micrchip 32bit processor
- Yun - Arduino Yun
- Other - configure your own board

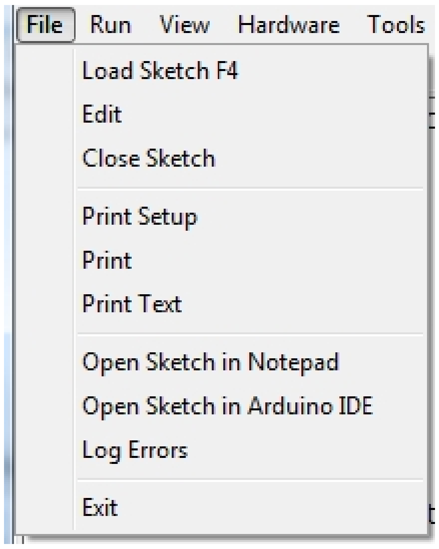
Menu System



The Menu system has eight sub-menus. These menus allow the Simulator to be customised.

Note that right clicking in the toolbar will bring up a Language popup menu.

File Menu



The File Menu Items are:

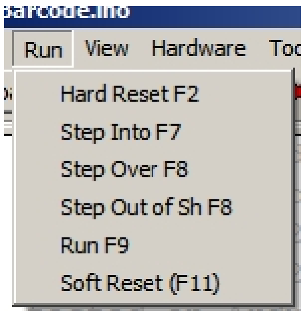
- Load Sketch F4 - Load a Sketch into the Program Window
- Edit Sketch F6 - Edit a Sketch in the Edit Window
- Close Sketch - close the sketch so on startup Simulator is empty

- Print Setup - open the Printer Setup Dialog Box
- Print - print a scaled picture of the program
- Print Text - print the sketch as a text file with no formatting

- Open Sketch in Notepad - this opens the sketch in Notepad for quick viewing and editing of the sketch
- Open Sketch in Arduino - this opens the sketch in the Arduino IDE if Arduino is the default program to open sketches. If this does not happen, right click a .ino sketch, select Open With and then Choose Default Program and set to Arduino.exe
- Log Errors - all errors will be logged to the ErrorLog + date.txt file Note that Errors here mean program and simulator errors

- Exit - close the program

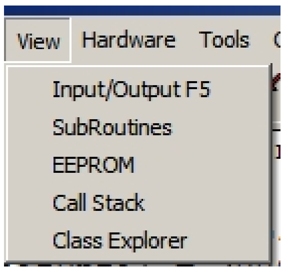
Run Menu



The Run Menu Items are:

- Hard Reset - reset the program normally
- Reset - reset the program to the start or Setup routine and clears all variables
- Step Into - step one line through the program, and step into any subroutines
- Step Over - step one line and step over any subroutines running the code inside the subroutine
- Step Out of - Step out of a subroutine
- Run - perform a step once every time period which is 5ms by default -see Shortcut toolbar
- Soft Reset - a new function to reset the sketch without needing to reload all the #include files

View Menu

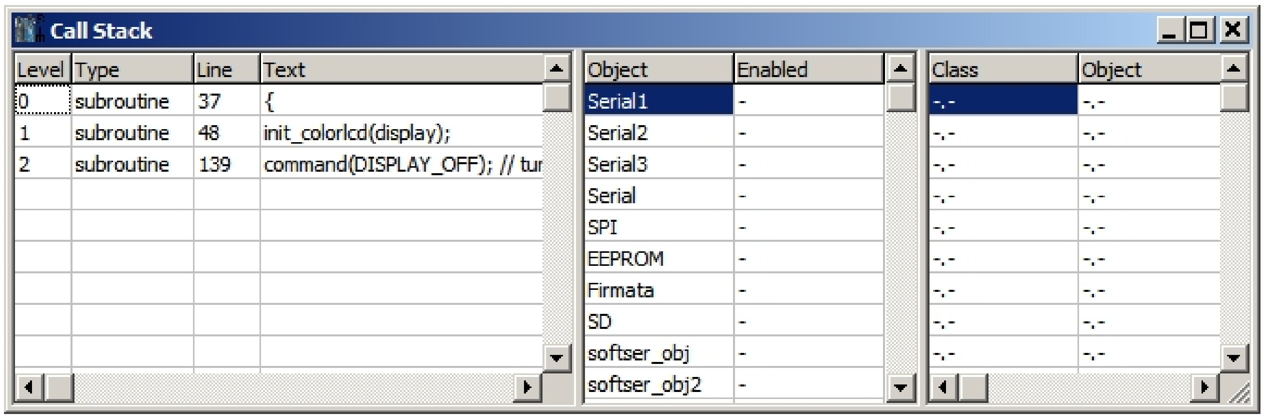


The View Menu Items are:

- Simulation data - open or Close the Simulation input/output window
- Subroutines - View the Subroutines and Line Numbers - double click on any subroutine to see it in the sketch window without changing the program counter
- EEPROM - View the EEPROM data
- Open the Call Stack Window
- Open the Class Explorer - view class objects with their values

The Call Stack Window is very important in debugging since it shows the origin of the current loop which may be a subroutine, while loop or any other condition. The loop level is show at the right in the Status bar.

The middle area is the standard Arduino library objects, while the right area is for user defined C++ objects.

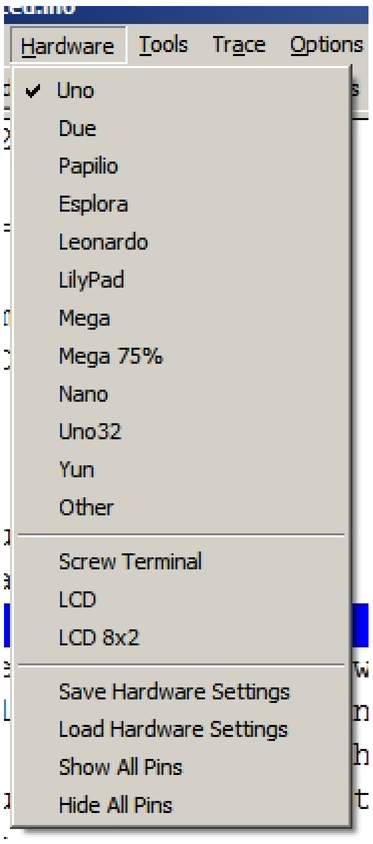

Call Stack

Level	Type	Line	Text
0	subroutine	37	{
1	subroutine	48	init_colorlcd(display);
2	subroutine	139	command(DISPLAY_OFF); // tur

Object	Enabled
Serial1	-
Serial2	-
Serial3	-
Serial	-
SPI	-
EEPROM	-
Firmata	-
SD	-
softser_obj	-
softser_obj2	-

Class	Object
-.-	-.-
-.-	-.-
-.-	-.-
-.-	-.-
-.-	-.-
-.-	-.-
-.-	-.-
-.-	-.-
-.-	-.-

Hardware



The Hardware Menu Items are:

- Uno - display the Board picture as an Arduino Uno board
- Due - display the Board picture as the ARM-based Arduino Due
- Papilio - display the Board picture as the Papilio One FPGA devkit
- Leonardo- display the Board picture as an Arduino Leonardo board
- LilyPad - display the Board picture as an Arduino LilyPad board
- Mega - display the Board picture as an Arduino Mega board
- Mega (x75%) - a smaller version for laptops
- Nano - display the Board picture as an Arduino Nano board
- Uno32 - display the Microchip Uno32 board
- Yin - display the Arduino Yun board
- Other - open up any picture file preferably 350 pixels wide to overlay on the Arduino picture

- Screw Terminal - Show the Virtronics custom Screw Terminal board
- LCD - Show the Virtronics custom LCD board with a 16x2 LCD fitted
- LCD 8x2 - - Show the Virtronics custom LCD board with a 8x2 LCD fitted

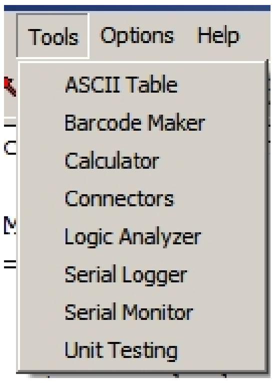
- Save Hardware Settings - save hardware settings to a .txt file which can be edited later
- Load Hardware Settings - load hardware settings from a .txt file
- Show All - Show all the pins, pin names and analog values for the Arduino board
- Hide All - Hide all the pins, pin names and analog values for the Arduino board

The hardware settings are saved to a text file with the format shown below. Clicking anywhere on Arduino picture will show the X,Y coorindates in the Status Bar and also save them to the clipboard.

```
Picture(Uno,350,506) // pix,width,height
resetSw(345,105) // Reset Switch x,y
onLed(273,454) // power led x,y
```

txLed(276,256)// transmit led x,y
rxLed(260,256)// receive led x,y
d13Led(309,256)// d13 led x,y
crystal(192,160)// crystal x,y
usb(250,32) // usb x,y
maxPin(13,5,19)// maxDigitalPin,maxAnalogPin,maxAllPins
externalInterrupts(2,3) // External Interrupts int0pin int1pin ...
LCD_Enabled(-29,-16,200,200,0,0,0,0,0)// x,y,w,h,pixel w, p-p,blit_color,oncolor,offcolor width=6*lcdx*p-p height=10*lcdy*p-p - for graphic screen
digitalPin0(367,490,340,490,396,490) // pin x,y pin label x,y analogValue x,y
...
analogPin0(45,406,71,406,2,406) // analog label x,y pin x,y analogValue x,y
...

Tools



There Menu Tools are:

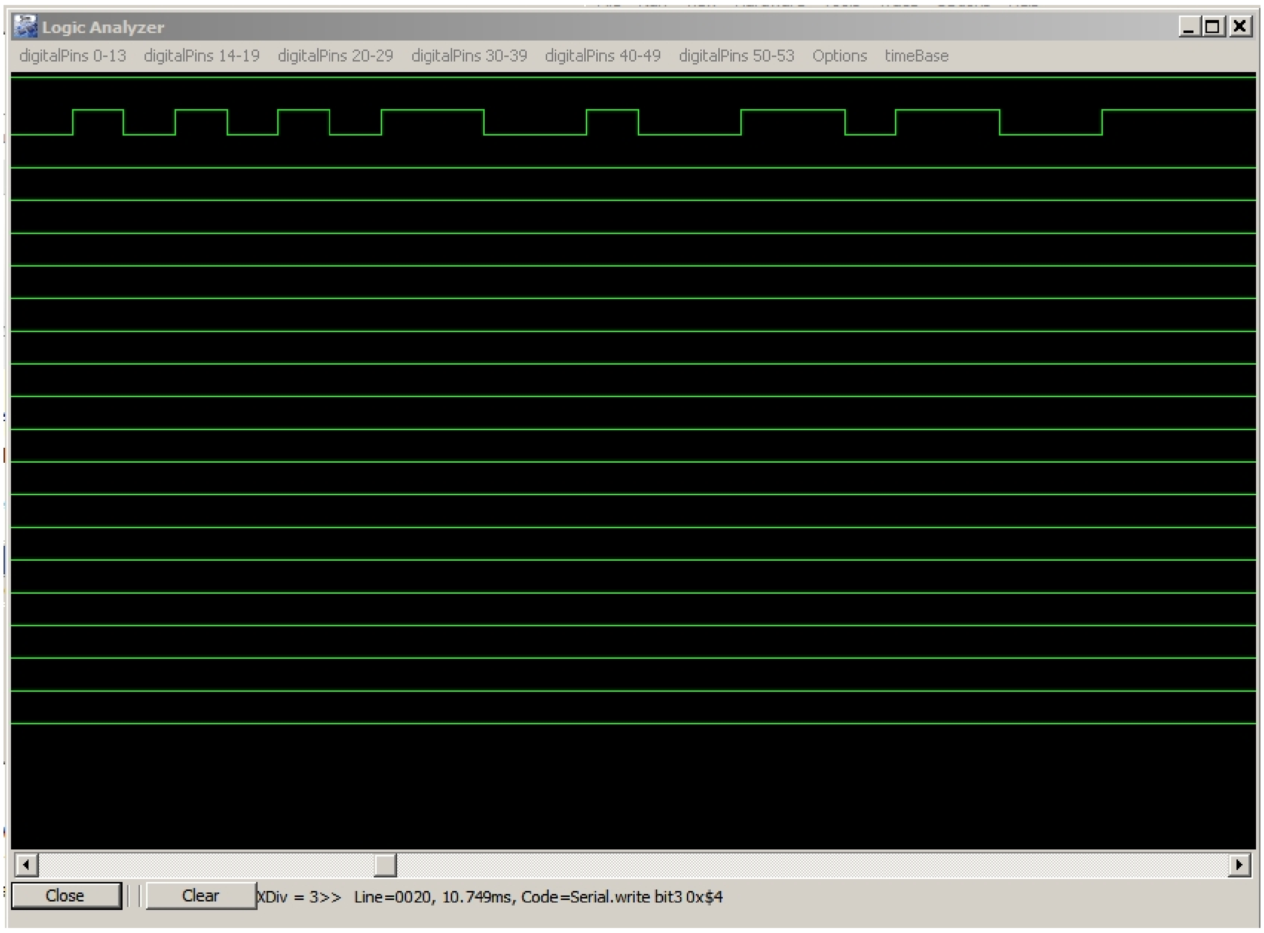
- ASCII Table - view the full ASCII table with character, hex and binary representations. Useful to most programmers and now viewable with large Text.
- Barcode Maker - the ability to set and print various barcodes - useful for the Virtronics Barcode Scanner Shield
- Calculator - a simple programming calculator for adding and multiplying hex numbers
- Connectors - view some common electrical connectors such as the serial D9 connector, USB typeA, USB mini and DC jack
- Logic Analyzer - view digital pins 0-13 graphically using the trace - automatically activates the File|Trace option
- Serial Logger - show Serial data graphically from the real World or from the Simulator
- Serial Monitor - an improved Serial monitor with a classic Bluegiga toolbar
- Automated Testing - the ability to automatically test series of sketched or commands. Very useful.

In all these Tools (except Automated Testing), the keyboard commands such as F7 to single step and F8 to step one line at a time will work.

Logic Analyzer

The Logic Analyzer can show the pin states in a graphical format using the trace data.

The menu allows pins to be turned on and off, and scrolling the mouse wheel can expand or shrink the time scale.



XDiv = 3>> Line=0020, 10.749ms, Code=Serial.write bit3 0x$4

Options

- All Pin On - turn on all 53 pins
- All Pins Off - turn off all 53 pins
- Pins 0-13 On - turn on the Arduino digital pins only
- Pins 0-19 On - turn on the Arduino digital and Analog pins (analog pins 0-5 map to pins 14-19)

- Auto Scale Y - allow the waveform to use the vertical space
- Load - load an old trace file
- Enable Green line - displays a green line inteh sketch window corresponding to where the prorgam was at that stage

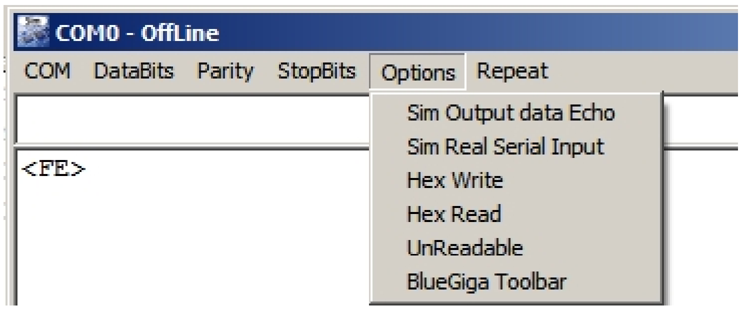
- Debug - turn on a whole stack of debug data (may be removed at a later date)

TimeBase - turn on vertical lines for the selectd timebase

Serial Monitor

The Serial Monitor is provided to be able to view real Serial Data.

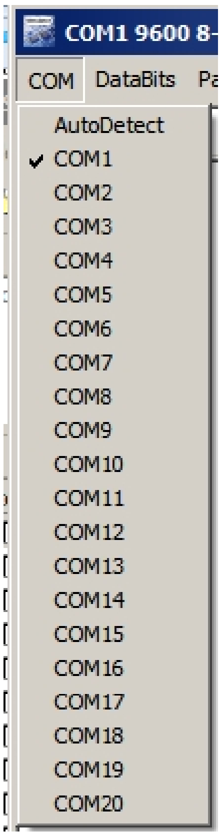
The Serial Data will be transmitted from the Simulator out the designated Serial Port.



The Options are:

- Sim Output data Echo - allows for data transmitted using the Serial.write or Serial.print commands to be Echo in the transmit box
- Sim Real Serial Input - great feature to allow incoming serial data to be ported to the Serial Indata in the Input/Output window
- Hex Write - send outgoing data as two hex characters followed by a space
- Hex Read - display incoming data in Hex format
- Unreadable - show unreadable characters as their character representation otherwise display as <hh> where hh is the hex code
- BlueGiga Toolbar - for those using classic Bluegiga device such as the ones from ESDN

In the COM Menu item, select AutoDetect to view only the available Com ports.



Unit Testing

Automated testing is an extensive page setup to automatically test sketches against set outputs. This helps with testing the Simulator, and while it takes a lot longer to setup a test script, in the long term, this will save much time.

At the bottom are 4 buttons labelled T1 to T4 and new. These are for routine tests which are all text files found in the _Sim_Test folder:

- T1 is for the Examples from 1.Basic up to and including ArduinoISP
- T2 is for the rest of the Official Arduino examples using the standard libraries such as SPI, Wire and Ethernet
- T3 is for bug fixes and known previous issues
- T4 is for a Simulator-oriented test script test each function as it is listed inside the Simulator source code
- New shows the latest test scripts added

Each line in a script file conforms to the following basic syntax:
test(function or variable, expected integer result expected float result, expected string result)
NA means not applicable or No test

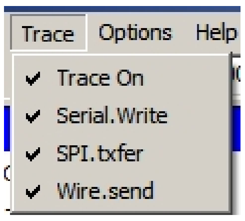
Extra single letter functions are:

- + for single step
- @for read data from Input/Output port using the integer result as the line number (SoftSerial,Serial,SPI,EthernetClient,EthernetServer,Ethernet, Wire,LCD, KeyBoard,File & Udp)
- ! load a file
- ~digPin - read a digital pin state with the pin set by the integer result and check the colour is the same as the expect float result read as an integer

The analog(x,val,NA,NA) can only be used once at the start of the script

```
Examples(Int,Float,Str,Output)
analog(0,123,NA,NA)

// ***** 0 BASIC *****
test(!01.Basics\AnalogReadSerial.ino,NA,NA,NA) //  1.1 AnalogReadSerial
test(+++sensorValue,123,NA,NA)
test(++@Serial,0,NA,123)

test(!01.Basics\Blink.ino,NA,NA,NA) // // 1.2 Blink.ino
test(+~digPin,13,0xffffff,NA)
test(++~digPin,13,255,NA)
test(+millis() ,1000,NA,NA)
test(+~digPin,13,0xffffff,NA)
test(+millis() ,2000,NA,NA)
```

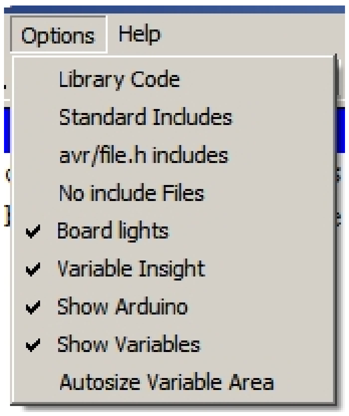
Trace



- Trace On - save all program steps to a text file with the filename based on the current date. the digital pins state is also saved for the Logic Analyzer
- Serial.Write - trace into all the output bits of each serial.write instruction and add time to the millis function depending on the baud rate
- SPI.txfer - trace into the output bits of each 8 bit transfer and add 24us for each transfer (3us for each bit)
- Wire.send - trace the output bits of each I2C output byte

The middle three options are specially designed for the Tool > Logic Analyzer and allow the digitalPin changes to be view in real time similar to the way an oscilloscope would display the voltage levels. When Trace On is selected or deselected, the lower 3 options are also selected or deselected.

Options



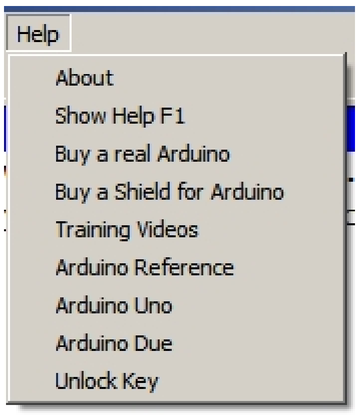
The Option items are:

- Library Code - open and display the actual code for a library such as <SPI.H> - note the Library director should be setup to point to the right directory first - see here for more info.
- Standard Include - open and display the actual code for header files such as <stdint.h> and <inttypes.h>
- avr.file.h - open and display the actual code for any avr/file.h files.
- No Include Files - stop those annoying open file dialogs - may not allow classes to be used.
- Board Lights - Uncheck to be able to run graphic colour screen sketches without the board leds activating
- Variable Insight - Uncheck to turn off the sketch window hover function to see variable value during stepping
- Show Arduino - Uncheck to hide the Arduino picture
- Show Variables - Uncheck to hide the Variables area
- Autosize Variable Area - automatically resize the variable area when reset or a new sketch is loaded

Please note that having any of the options turned on such as the include and library code options can cause many errors since these files use high level C++ syntax structures so these options should be used with care.

On the positive side, these libraries allows for library code to be easily inspected. This can be very useful when adding a colour display from an online store such as Adafruit and then being able to open the Adafruit_GFX and Adafruit_SSD1206 library to see how the SPI port drives the colour display.

Help



The Help Menu items are :

- About - Show the Program version
- Show Help F1 - show this help - Note: this can also be activated by pressing F1
- Buy a real Arduino - open the Buy Arduino page
- Buy a Shield for Arduino - open the Shields for Arduino page
- Training Videos - watch all the Youtube videos related to the Simulator and Shields
- Arduino Reference - display the Arduino language reference page
- Arduino Uno - display the Arduino Uno Page
- Arduino Due - display the Arduino Due Page
- Unlock Key - view the Unlock Key and licence info - please note the Pro Version now requires an Unlock

The Help | About screen shows the current software version.

The F (for Free version) after the version number indicates the Simulator is identical to the Pro Version but . The Free version has been provided as demo version to allow the Simulator for Arduino program to be evaluated prior to purchase. It is anticipated that after a month of use, it will be more economically viable to purchase the Pro Version than continue to run the Free Version.

Limitiations of the Free version are:

- a Code limit of 150 lines which will be displayed on the Delay Timer screen
- an incrementing delay timer (from 30 seconds) window any time a new sketch is loaded or edited
- A trial period of 30 days or 10 sketches with no timer after an unlock process
- An initial trial period of 30 minutes (no unlock required)

Dialog Windows

The Simulator has several dialog windows which can display more information when activated. These are:

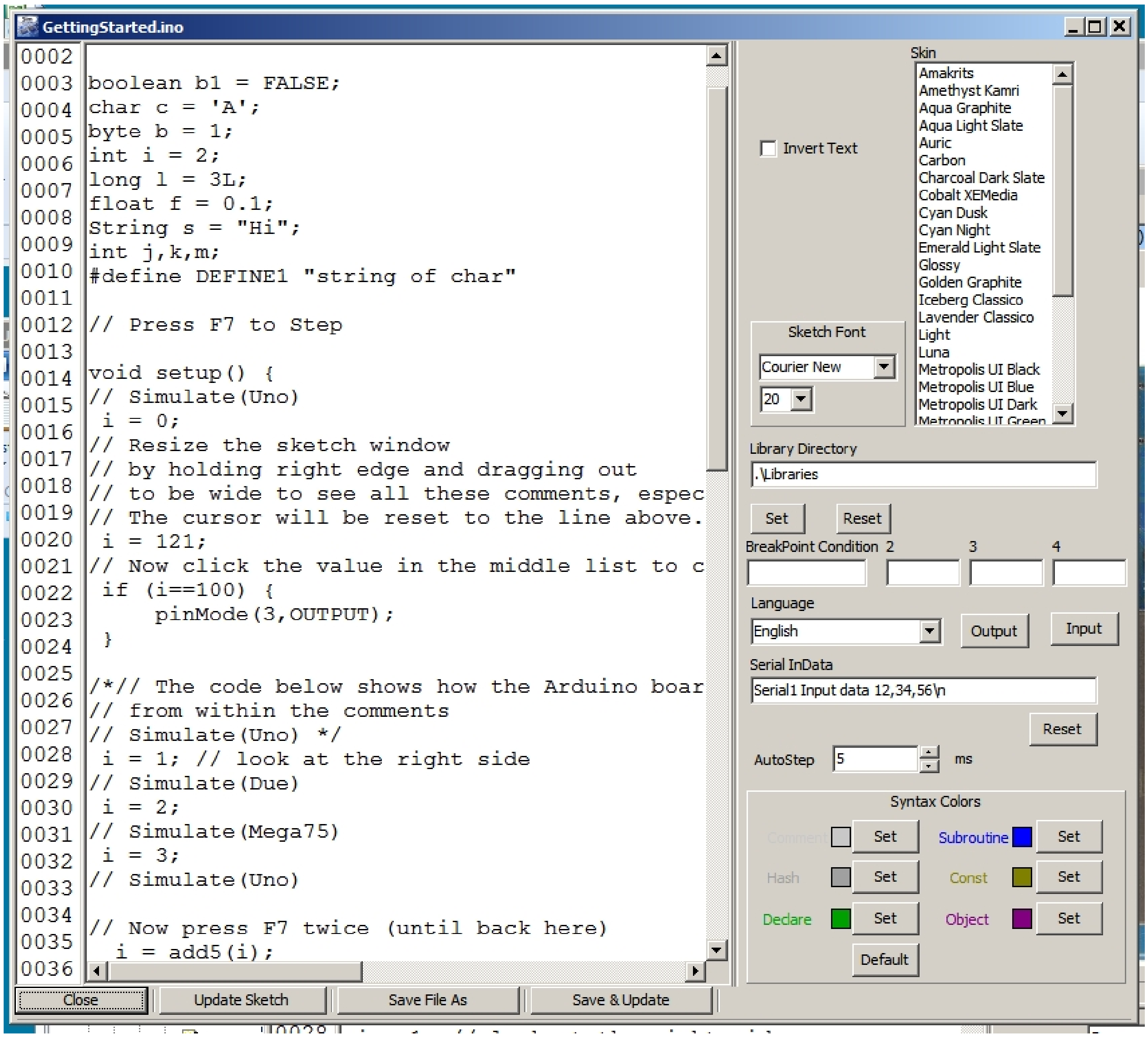
- The Edit Window
- The Input/Output Window
- View Subroutines
- Error Message

Edit Window

The Edit Sketch Window allows for sketches to be edited and updated or reloaded into the Program window

- Close - closes this window and resumes simulating the program
- Update Sketch - save this edited sketch back to the program window
- Save File As - save this sketch to a new .ino file
- SaveUpdate - save the sketch under the same name and update the sketch in the program window
- Bare Minimum - load the bare minimum code skeleton - see here for more info

- Sketch Font type - changes the font type of the main program window
- Sketch Font size - changes the font size of the main program window
- Skin - this will change and save the skin for the whole Simulator program. The default skin is Windows. Click Invert to show text against a dark background
- Library Directory - set for using custom libraries or click Reset to set the Library directory to the default value
- Language - change between English, French, Italian and Other for titles only - also Input and output the language text
- BreakPoint condition - set to break on a certain condition at the red line. The condition can be any expression and can even be a formula - such as i+=10.
- Serial InData - provides the ability to Change the Default Serial - Also available as a hidden code command - Reset to set default value of 123456ABCDEF\n
- AutoStep - the delay between single steps in the run mode - set this to 1 or 0 to run without any refresh
- Syntax Colours - allows the user to adjust and save the syntax colours - Default will reset all colours


GettingStarted.ino

```

boolean b1 = FALSE;
char c = 'A';
byte b = 1;
int i = 2;
long l = 3L;
float f = 0.1;
String s = "Hi";
int j,k,m;
#define DEFINE1 "string of char"

// Press F7 to Step

void setup() {
// Simulate(Uno)
 i = 0;
// Resize the sketch window
// by holding right edge and dragging out
// to be wide to see all these comments, espec
// The cursor will be reset to the line above.
 i = 121;
// Now click the value in the middle list to c
 if (i==100) {
     pinMode(3,OUTPUT);
 }

/*// The code below shows how the Arduino boar
// from within the comments
// Simulate(Uno) */
 i = 1; // look at the right side
// Simulate(Due)
 i = 2;
// Simulate(Mega75)
 i = 3;
// Simulate(Uno)

// Now press F7 twice (until back here)
  i = add5(i);
```

Skin: Amakrits, Amethyst Kamri, Aqua Graphite, Aqua Light Slate, Auric, Carbon, Charcoal Dark Slate, Cobalt XEMedia, Cyan Dusk, Cyan Night, Emerald Light Slate, Glossy, Golden Graphite, Iceberg Classico, Lavender Classico, Light, Luna, Metropolis UI Black, Metropolis UI Blue, Metropolis UI Dark, Metropolis UI Green

☐ Invert Text

Sketch Font: Courier New, 20

Library Directory: .\Libraries

BreakPoint Condition 2 3 4

Language: English

Serial InData: Serial1 Input data 12,34,56\n

AutoStep: 5 ms

Syntax Colors: Comment ☐, Hash ▨, Declare 🟩, Subroutine 🟦, Const 🟫, Object 🟪

Bare Minimum

Bare Minimum will load the bare minimum needed for a new sketch. Select Run|Edit Sketch and the press the rightmost button to load the Bare Minimum to start writing a sketch. The text will appear as shown below. Modify and then click the button Save File As to save the sketch to a new filename.

```
// Simulate(Uno) link sketch to Arduino board here

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

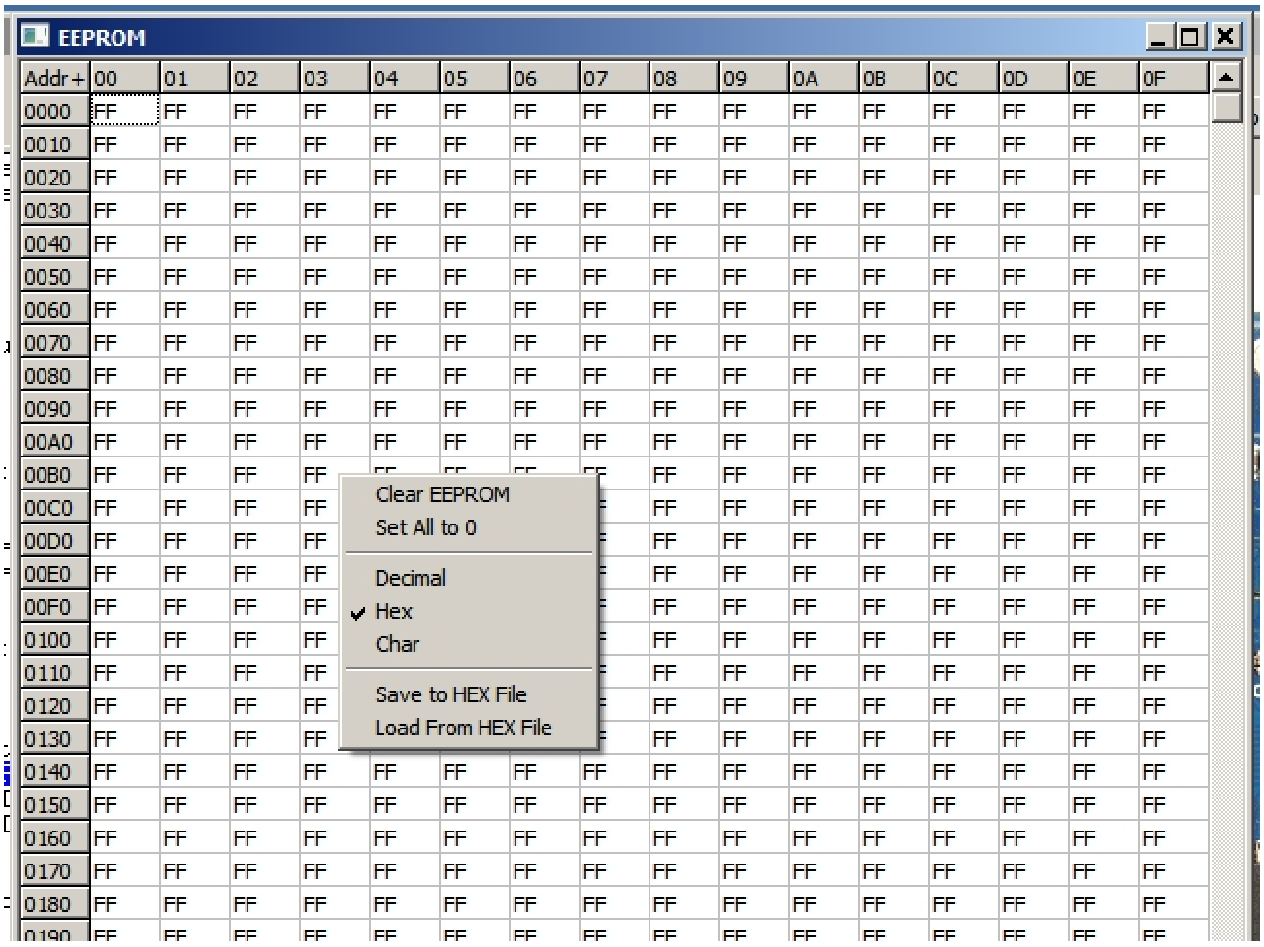
EEPROM Window

The EEPROM window allows for viewing and modifying of the EEPROM memory.

The EEPROM Window also has the added function of being able to view a full hex file contents, the ability to view the contents in different formats (eg Decimal, Hex or Char) and being able to change and save the file. there are many hexfile editors available on the internet and this feature is just a small part of the Simulator

Right click to bring up the popup menu which has several options:

- Clear EEPROM - set all EEPROM memory to FF
- Set All to 0 - all EEPROM memory to 00
- Decimal - display Variables value in decimal
- Hex - display Variables value in hexadecimal
- Char - display Variables value in Character format and hexadecimal
- Save to Hex File - save the EEPROM memory to a hex file
- Load from Hex File - load the EEPROM memory from a hex file



Addr+	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00C0	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
00D0	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
00E0	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
00F0	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
0100	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
0110	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
0120	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
0130	FF	FF	FF	FF					FF	FF	FF	FF	FF	FF	FF	FF
0140	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0150	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0160	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0170	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0180	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0190	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Input/Output Window

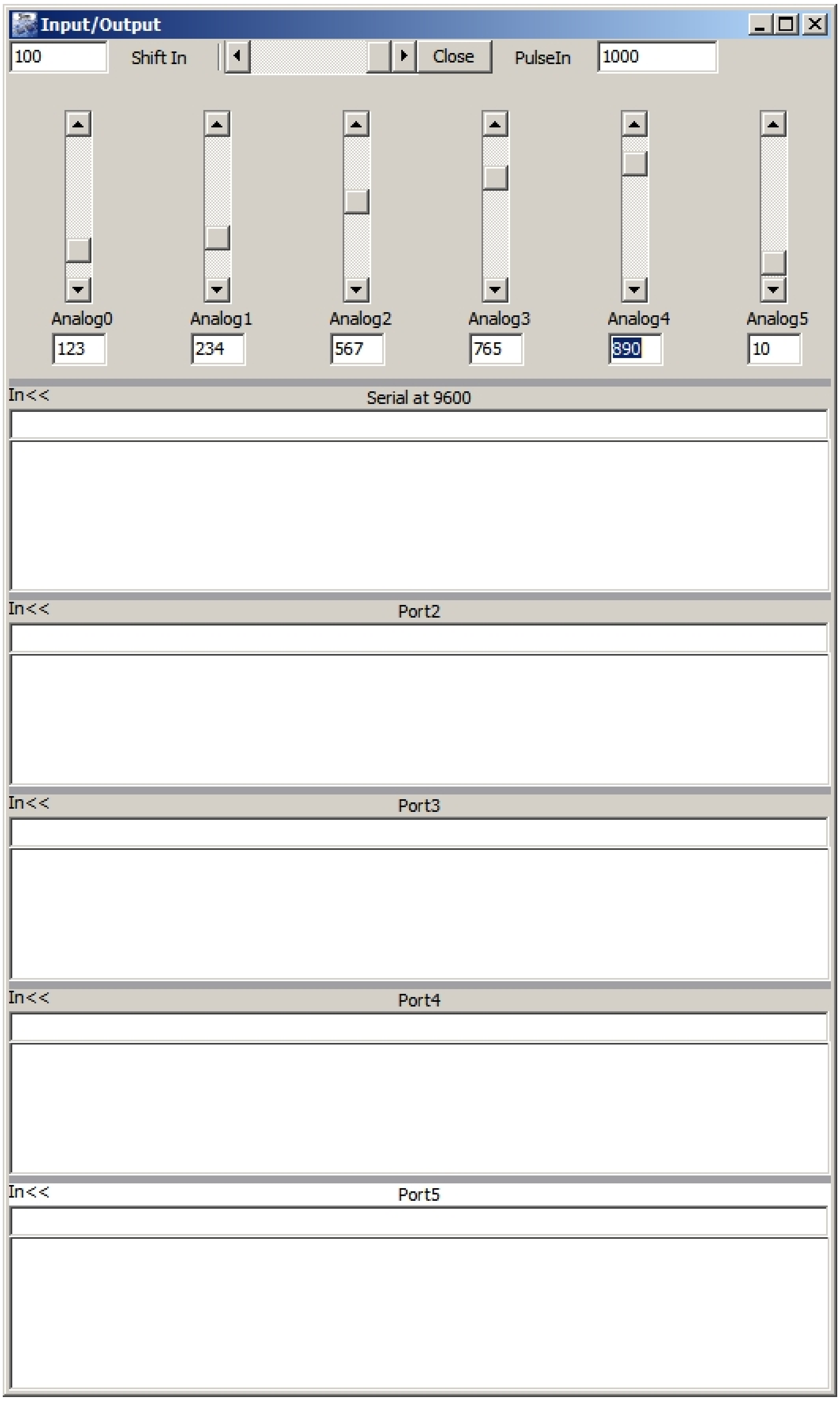
The Simulation Window allows for data to be simulated for the six analog inputs for the Uno or 16 analog inputs for the Mega. The window may be resized to be smaller, and the scrollbars and text boxes will automatically resize to suit.

The ShiftIn and PulseIn allow the return values for these functions to be set.

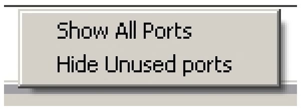
The Analog Value may be changed by moving the respective scrollbar, or entering a new value in the white Edit box.

Input and Output data is displayed in five ports. Each port has its own line for input data and a box for output data. Each Port can be expanded or contracted to suit the display and resized.

The ports are assigned in the order they are setup.

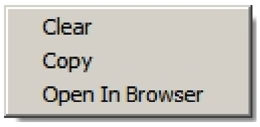


Simulation Popup Menu



The Simulation Window Popup Menu has two items:

- Show All Ports - to view the five ports and reset the Splitter bars if needed
- Hide Unused Ports - close all ports except the opened ports

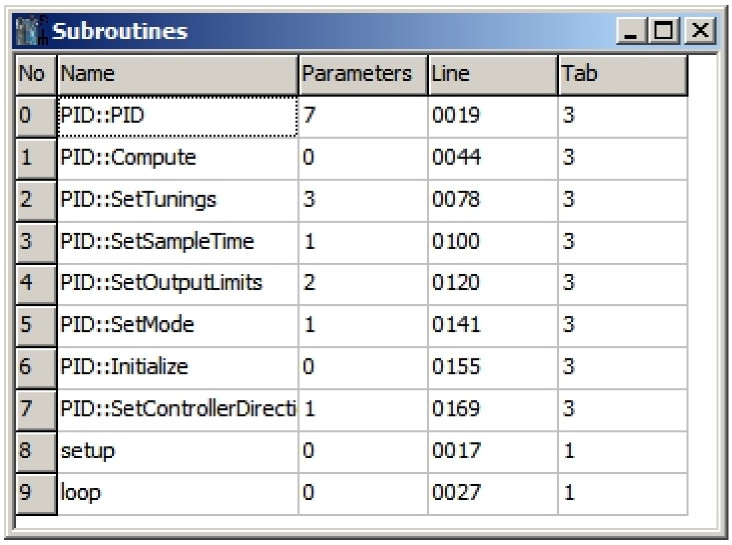


Inside each port is another Popup window with three options:

- Clear - clear the Port screen
- Copy - copy the data to the clipboard
- Open in Browser - open the Port data in the default web browser, useful for sketches using Ethernet

View Subroutines

The View Subroutines window allows for the Subroutines to be viewed along with the number of parameters, the line number and the tab. The line numbers may be viewed in the program window by selecting Line Numbers



No	Name	Parameters	Line	Tab
0	PID::PID	7	0019	3
1	PID::Compute	0	0044	3
2	PID::SetTunings	3	0078	3
3	PID::SetSampleTime	1	0100	3
4	PID::SetOutputLimits	2	0120	3
5	PID::SetMode	1	0141	3
6	PID::Initialize	0	0155	3
7	PID::SetControllerDirecti	1	0169	3
8	setup	0	0017	1
9	loop	0	0027	1

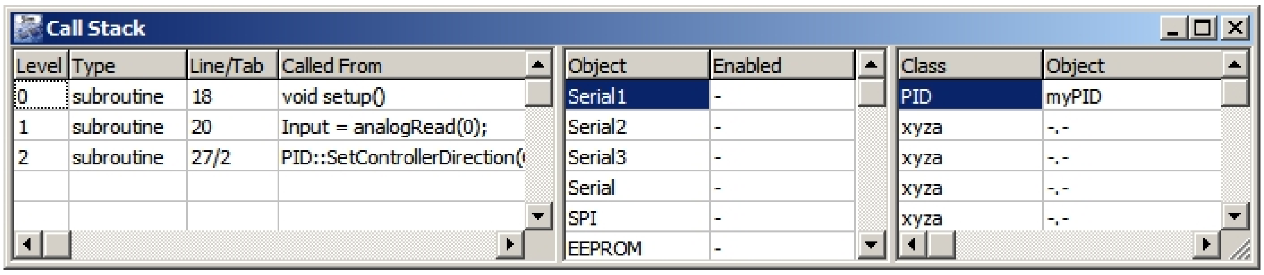
There is now a new Combo Box on the Shortcut toolbar which allows for all the subroutines to be viewed and shown in the Program Window without changing the program counter.

Call Stack

The Call Stack displays the different levels of sketch loops in the left grid

The middle grid shows the Arduino objects, names and Enabled status.

The right grid shows classes found and objects derived from these classes.



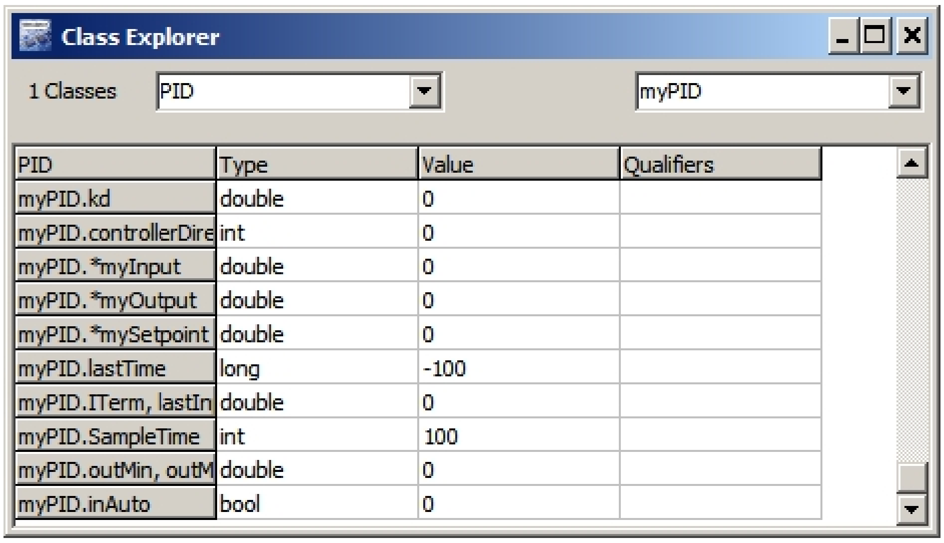
Call Stack

Level	Type	Line/Tab	Called From
0	subroutine	18	void setup()
1	subroutine	20	Input = analogRead(0);
2	subroutine	27/2	PID::SetControllerDirection(

Object	Enabled
Serial1	-
Serial2	-
Serial3	-
Serial	-
SPI	-
EEPROM	-

Class	Object
PID	myPID
xyza	-.-
xyza	-.-
xyza	-.-
xyza	-.-

Class Explorer



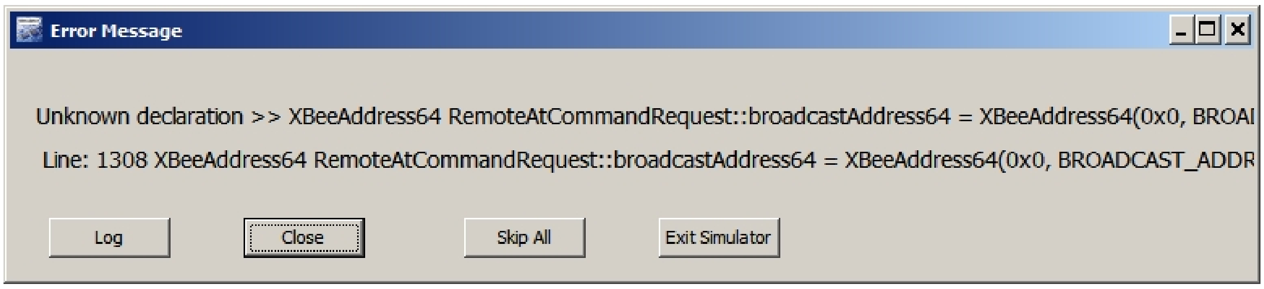
Class Explorer

1 Classes PID myPID

PID	Type	Value	Qualifiers
myPID.kd	double	0	
myPID.controllerDire	int	0	
myPID.*myInput	double	0	
myPID.*myOutput	double	0	
myPID.*mySetpoint	double	0	
myPID.lastTime	long	-100	
myPID.ITerm, lastIn	double	0	
myPID.SampleTime	int	100	
myPID.outMin, outM	double	0	
myPID.inAuto	bool	0	

The Class Explorer allows for all the class variables to be viewed and inspected. These may only be modified in the Variable Area at present.

Error Message



Error Message

```
Unknown declaration >> XBeeAddress64 RemoteAtCommandRequest::broadcastAddress64 = XBeeAddress64(0x0, BROAI
Line: 1308 XBeeAddress64 RemoteAtCommandRequest::broadcastAddress64 = XBeeAddress64(0x0, BROADCAST_ADDR
```

The Error message screen displays any code which the Simulator cannot process or which the Simulator decides if an illegal operation such as:

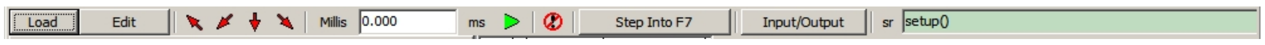
- the above operation, outside the setup() and loop() routines
- setting a digital pin which is set to input
- changing a const(ant) value
- setting a variable to a value outside its limits
- addressing an array element before the start or past the end of the array

The File|Log Errors option allows these errors to be logged to an ErrorLog +date.txt file.

The Log button will only be shown if the Error could be related to internal Simulator logic. If the Error message starts with Sketch Error, the Simulator has guessed that it is more likely the error is in the Sketch. If this is not the case, please email the sketch to support@virtronics.com.au

Shortcut Toolbar

The Shortcut Toolbar provides single click actions for commonly used operations. Right Click on this toolbar to show the Language popup menu which allows easy switching between English, French, and Italian(and other languages when requested).



Button Functions:

- Load - loads a sketch from the last used folder into the Simulator
- Edit - Edit a sketch

- Left up arrow - F2 Reset
- Left Down Arrow - Shift F8 Step out of subroutines excluding setup and loop
- Down Arrow - F8 Step Over - single step jumping over any subroutines
- Right Down Arrow - F7 Single Step and Step into any subroutines

- Millis - the run time in milliseconds (NOTE: ms not seconds). This value can be changed at any time. Each program step takes around 1us

- Run - F9 AutoStep the program one instruction every xx ms The ms value can be edited. This time is the delay between line execution and is not the actual speed of the program since each line may take 1-100ms to execute. If the AutoStep time period is set to 1ms, the variables, blue program traceline and statusbar will not update to increase speed.
- Abort - abort if the program is stuck in any loop

- Simulation - show or close the Simulation input/output window
- Step Into (F7) - this button allows for the program to be single stepped

- SubRoutines - contains a list of subroutines and shopws the current subroutine- select a subroutine from the list to jump and highlight in colour that subroutine first line without affecting the program counter

The Milli seconds variable is displayed in a text box. This value can be adjusted by typing in a new value. Each line of program execution takes 1us or 0.001. The commands Delay and delayMicroseconds will add the accurate number of milliseconds or microseconds to the Millis Text box.

Shortcut keys

Several keyboard shortcuts have been added. These are

- F2 for Reset
- F3 for Find
- F4 for Load a sketch
- F5 to open the Simulation Window
- F6 - Edit a sketch in the Edit Window
- F7 to Step Into
- F8 to Step Over
- Shift F8 to Step out of a subroutine
- F9 to Run or AutoStep
- Ctrl-F for Find (same as F3)
- F11 for Soft Reset

Example Sketches

Several sketches have been included for testing the Simulator. These are the Official Arduino examples in the folders 01 Basics to 19 Wire.

In addition there are:
_Sim_Samples - Samples for the latest Youtube videos
_Sim_Shields - sketches to run on the custom Virtronics Shields for Arduino
_Sim_Test - test sketches and sketches sent in for error checking and fixing
_SIM_Extra - sketches with some new tryout features and sketches from the Videos

Libraries - this folder has some test Library sketches and various header files

In a typical install, the sample sketches will be found in this folder:
C:\Program Files\Virtronics

or for Windows 64 bit
C:\Program Files (x86)\Virtronics

Test Sketches

The test sketches are made up of issues with the Simulator and are used to do quick checks on patch versions of the Simulator. The major releases of the Simulator are fully tested against most of the sample sketches.

The test Sketches can be found in the _SIM_Test subfolder.

Language Processing

Keywords

The following words are defined as keywords and should not be redefined in a sketch:

defined(ARDUINO)	1
defined(__AVR__)	1
defined(__AVR_ATMEGA168__)	1 for Uno or 0 for Mega
defined(NUM_ANALOG_INPUTS)	6 for Uno or 16 for Mega

FALSE	0
TRUE	1
LOW	0
HIGH	1
ARDUINO	100
TOTAL_ANALOG_PINSX	// unused - default return value of 1
TOTAL_PORTS	14
INPUT	0
OUTPUT	1
INPUT_PULLUP	2
TOTAL_PINSX	20
SD_CARD_TYPE_SD1	1
SD_CARD_TYPE_SD2	2
SD_CARD_TYPE_SD3	3
PI	3.1415926535897932384626433832795
HALF_PI	1.5707963267948966192313216916397
TWO_PI	6.283185307179586476925286766559
DEG_TO_RAD	0.017453292519943295769236907684886
RAD_TO_DEG	57.295779513082320876798154814105

NUM_ANALOG_INPUTS	6 for Uno or 16 for Mega
TCC	0
NULL	0
DAC0	66
DAC1	67
ARDUINO	100
__AVR__	1
__AVR_ATmega328P__	1 or 0
__AVR_ATMEGA168__	0
NUM_ANALOG_INPUTS	6 or 16
__AVR_ARCH__	100
__AVR_ENHANCED__	0
false	0
true	1
__GNUC__	4
__GNUC_MINOR__	3
__INT_MAX__	6
RASPBERRY_PI	0
WL_IDLE_STATUS	0
WL_CONNECTED	1
WL_NO_SHIELD	-1
MAX_SOCK_NUM	4

PINA-L (no PINI)
DDRA-L (no DDRI)
PORTA-L (no PORTI)
defined(...)

Custom Libraries

The Simulator now has the ability to load custom libraries and single step through the code. Please note that pointers are not supported and this is the typical reason for a library not to run.

To use a custom library, there are two methods. The first is to copy the header file and cpp file to the same directory as the sketch, and then the Simulator will find these and load them in separate tabs.

The second method which is the preferred method is to use the library directory (.\Libraries) and then the Simulator will automatically find the library header and source file and load them in separate tabs in the Program Window. This method is better since the Simulator will find the files in the same place as the Arduino IDE, and will save the effort of copying the files. The Library Directory can be setup by pressing F6 to load the Edit Sketch page and at the right hand side is an Edit Box which allows the Library directory to be Set or Reset.

Millis

The Millis functions returns the number of milliseconds the program has been running for. In the Shortcut toolbar, the Millis edit box shows the value of milliseconds as a floating point number. The micro function will output this value * 1000.

Please note that the Simulator is not a real-time Simulator and runs around 1000 to 10000 times slower than a real Arduino.

Troubleshooting

- Menu has disappeared => resize the screen or right click on the Arduino picture and uncheck Minimize
- The Language is wrong => Click Edit (second from left top button) and select language and change

back to the selected language
- I have too many errors and just want to run my sketch => right click in the Sketch Window and uncheck Errors Enabled
- No Errors are appearing when they should => right click in the Sketch Window and check Errors Enabled
- An error has come up => click the Log button to log the error to Quality Central or Email to email the error with the sketch
- The Simulator doesn't work as expected => check the forum or email support@virtronics.com.au

Registry Settings

Disclaimer: Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows .

All the Simulator settings are saved in the registry under the key
HKEY_CURRENT_USER\SOFTWARE\SimForArduino x.xx where x.xx is the version number

The Registry settings can be viewed by running the RegEdit program or a similar utility.

Version Info

Version 1.10 Jan 2018
- fix micros
- and/or priority
- fix strcpy
- add ^= operator
- improve define includes - allow for #define(a,...)

Version 1.09 - skipped - no-one else does 9 releases - what happened to Windows 9?

Version 1.08 Oct 2017
- Allow up to 129 tabs now
- after reset, tabs and time to load now reported correctly
- allow for macro subsitution of object names - eg #define dbg Serial
- if an EEPROM object is found, there will be no error for <EEPROM.h> library not loaded
- add Timer1.read() function to read us from last interrupt
- fixup all .h and .c files loading including .html - now only .h,.hpp,.c,,cpp

Version 1.07 July 2017
- Fixup typedef enum
- fix long standing issue with long arrays >300 characters
- fixup array issue
- fixup static issue
- fixup return arguments

Version 1.06 Apr 2017
- Fixup negative enum values
- Fix nested case statements
- Allow more statements on top line of function
- fixup round function
- improve classes

Version 1.05 Jan 2017

- improve blue s/r box - make green
- fixup return bug
- improve callstack
- add TFT
- add SerialPassThrough to test
- add Wifi UDP
- add TFT - check 6 sketches
- Ethernet advanced Chat Server
- Add Adafruit playground - in progress
- fixup some issues in typedef and struct
- increase max loops to 300, args to 40

Version 1.04 - October 2016

- Improve blue s/r box
- add free version browser to timer
- improve resizing so sketch window always there
- bug fixes
- improve class explorer
- save Variables Dec/Bin/Hex on close
- save Input/Output window settings
- improve classes within classes xbee._response

Version 1.03 - June 2016

- fixup Windows 10 styles
- add in 8 more styles Vapor,Radiant,Jet,Emerald, Diamond, CopperDark, Copper, Coral
- fixup confusion between colon for labels and ?:
- fixup variables font
- save Edit font on Edit close and open
- fixup conversion from float to int
- print negative floats
- add in Check for Updates
- auto update silent check on startup
- reset free timer

Version 1.02 Mar 2016

1. Allow setup in file other than tab1
2. Allow loop in file other than tab1
3. allow for arduino format files in directory - open all files in directory
4. make App Analytics an option
5. fixup memory leaks
6. fixup variable insight
7. fixup spurious fn return values - clear on reset
8. Fixup analog write pins to align with analog read pins
9. fixup due analog pins
10. fixup barcode memory leak
11. Add recent
12. digitalRead now sets pin mode
13. Improve Class Explorer
14. Allow SD\CardInfo.ino to load with Uno with first 3 Options on
15. Show (ErrEn) in Status bar when Errors Enabled
16. Speed up Syntax Colouring change

Version 1.01 Dec 2015

1. Fixup up if else issue
2. Stop variables from flickering
3. Add alignment to pin names
4. Fixup the D13 light onthe Nano
5. Update Nano to have pin name alignment
6. Update Uno to have pin name alignment

7. Improve the Insert Tab option
8. Save file after insert tab
9. Fixup the match bracket feature
10. Update Sketch when changed externally

Version 1.00 Oct 2015

1. Keep variables spot on single step
2. Show conditional breakpoints on reset
3. Allow for Serial.begin(9600,SERIAL8N1)
4. Expand recursion from 40 to 60
5. allow 9999U as a valid unsigned constant
6. check and fix `val= analogRead(min(park,park0));`
7. fixup faulty breakpoint
8. fixup analogRead in Inputs.ino
9. fixup faulty arrays
10. various other fixes

v0.80 - v0.99G to June 2015 first test releases

Credits

This project has been a collaboration with many people.

Credit and thanks must go to these people for all their help: Adrian Wells, Marco Stuurman, Zeljko Frankovic, Pete Lunt, Mark Grass Sr, Serge Desjardins, Peter Brouggy, Halam Rose, Larry Vatland, Raimondas Butauskas, Jason Snow, J-Pierre Romanogli, Hamilton Elliott, Graeme Caie, Michael Moore, Filipe Oliveira, Leon Rozengarten, Robert Lopez,Mauro Abbattista,Todd Radack,Alain Herben, George Vrynios, Neko San, Mauro Abbattista, Alain Herben, Victor Aguilar, David Williams, Janis Gkatzaras, Rodrigo Amaya, Ed Ross, David Cox, Shane Stenton, Freddie Snijman, Andres j. Ogayar, Anxionnaz Yannick, Jeandaniel Planterose, Donald Dempsey, Enrique Condes, Peter Brown, Mladen Bruck, Jesse Carneiro, Nigel Woodford, Ken Jensen, Koen Victor,Damir Kudeljan,Maxwell Yun, Douglas Hendricks, Marco Baitelli, Howard Bassen, David Griffey, Patrick Beier, Fokko Dusseljee and few thousand more we will add when there is some spare time.