GadgetMaster Script Editor

Introduction

GadgetMaster Script Editor version 1.4 enables users to create simple scripts written using standard text. The main screen starts in the Script Builder tab, this screen has the script on the right and the motor and input / output status along the bottom. The Script Builder tab enables users to create a script simply by typing values such as the desired number of steps to turn the motor, then click 'Move'.

For example;
by entering 100 steps and selecting motor 1, the script
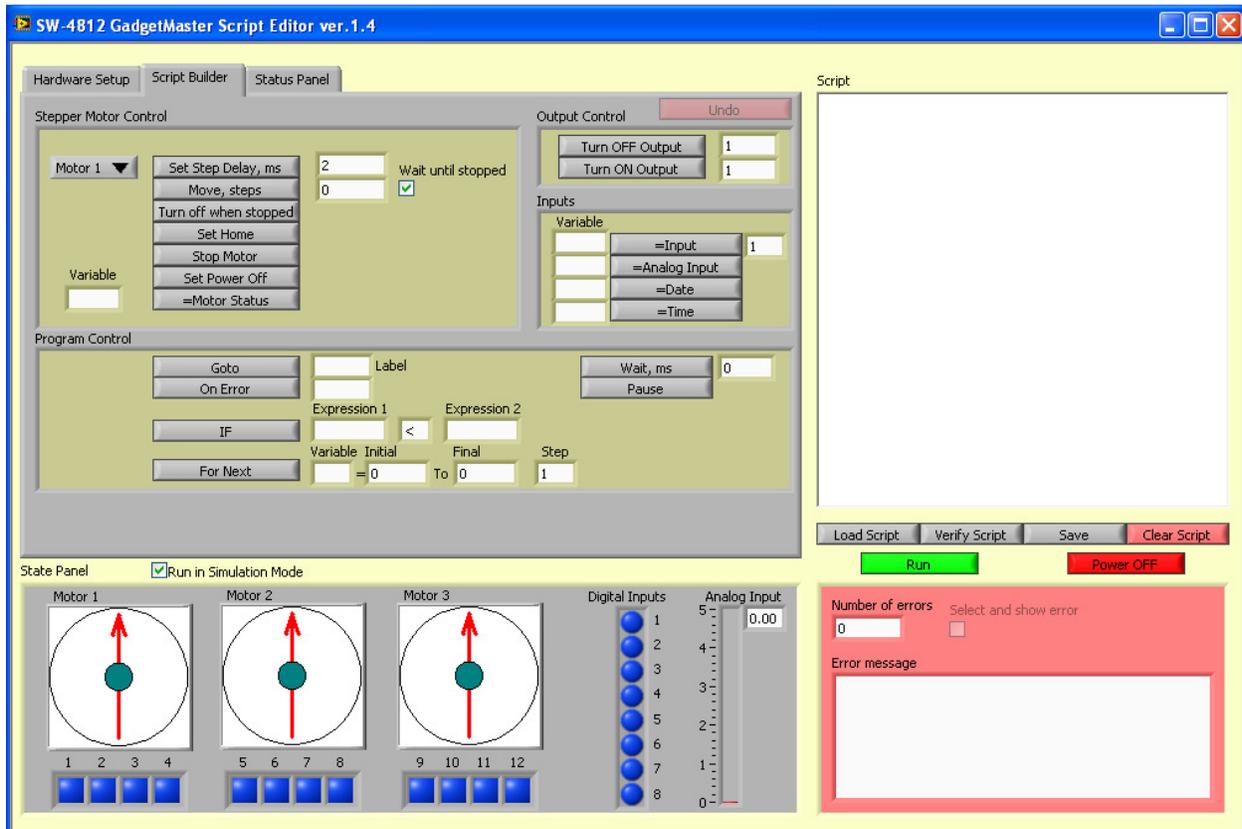Move(1) = 100
is automatically added to the script on the right.

Figure 1 Script Builder

Simulation Mode

By default the Script Editor is running in 'Simulation Mode' this allows the user to write and debug a script without connected any hardware. In this mode the Digital Inputs and the Analog Input can be adjusted by the user simply by clicking on the inputs or dragging up the value of the analog input.

Other Tabs

The first tab is the hardware tab. In the "Hardware Setup" tab the user selects the hardware configuration, such as the parallel port address, the jumper position and the motor connection. Usually, these parameters do not change during the execution of a script. There are, however, operators that allow the script to overwrite these selections so the program will use the new settings. Once the program is terminated the manual selection from the "Hardware Setup" tab takes effect.
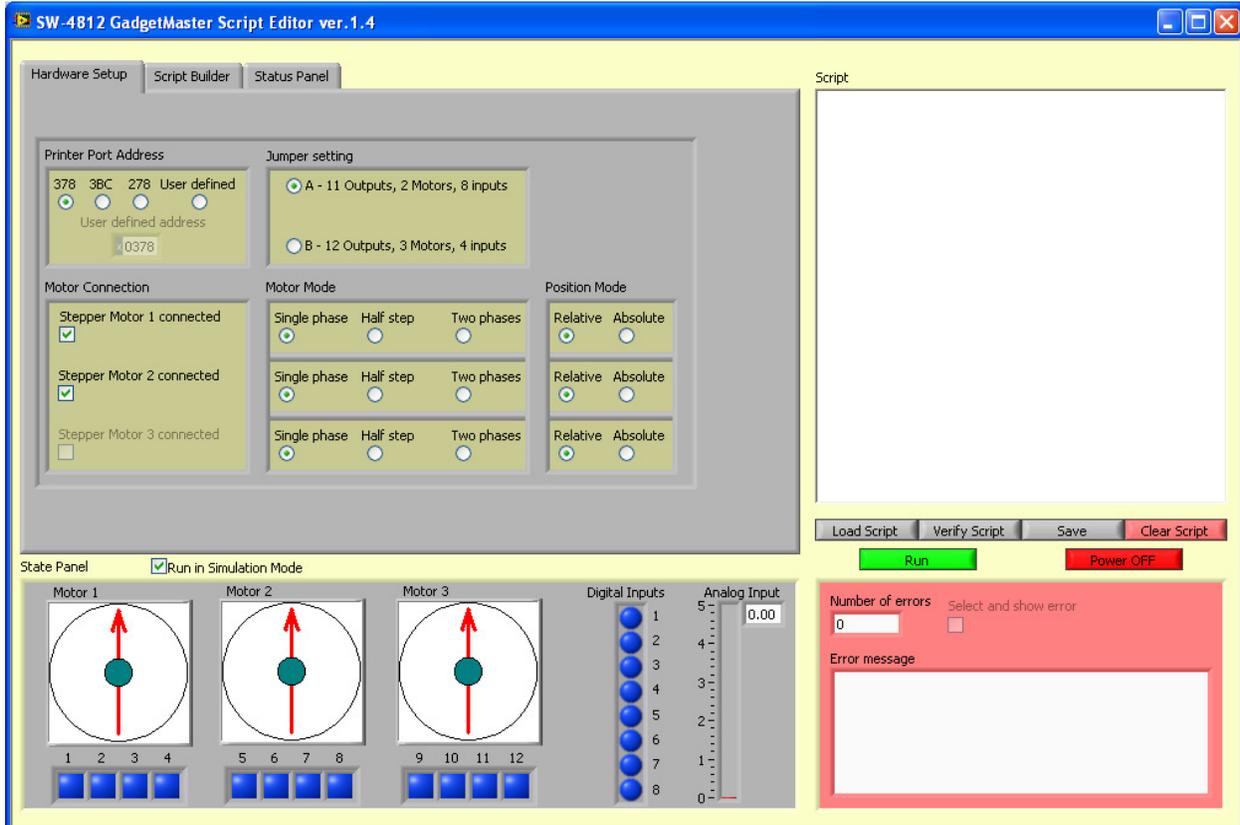

Figure 2 Hardware Setup

The final tab is the Status Panel. This tab automatically pops up when the script is running. It displays the status of the motors as well as the value of any variables that you have assigned.
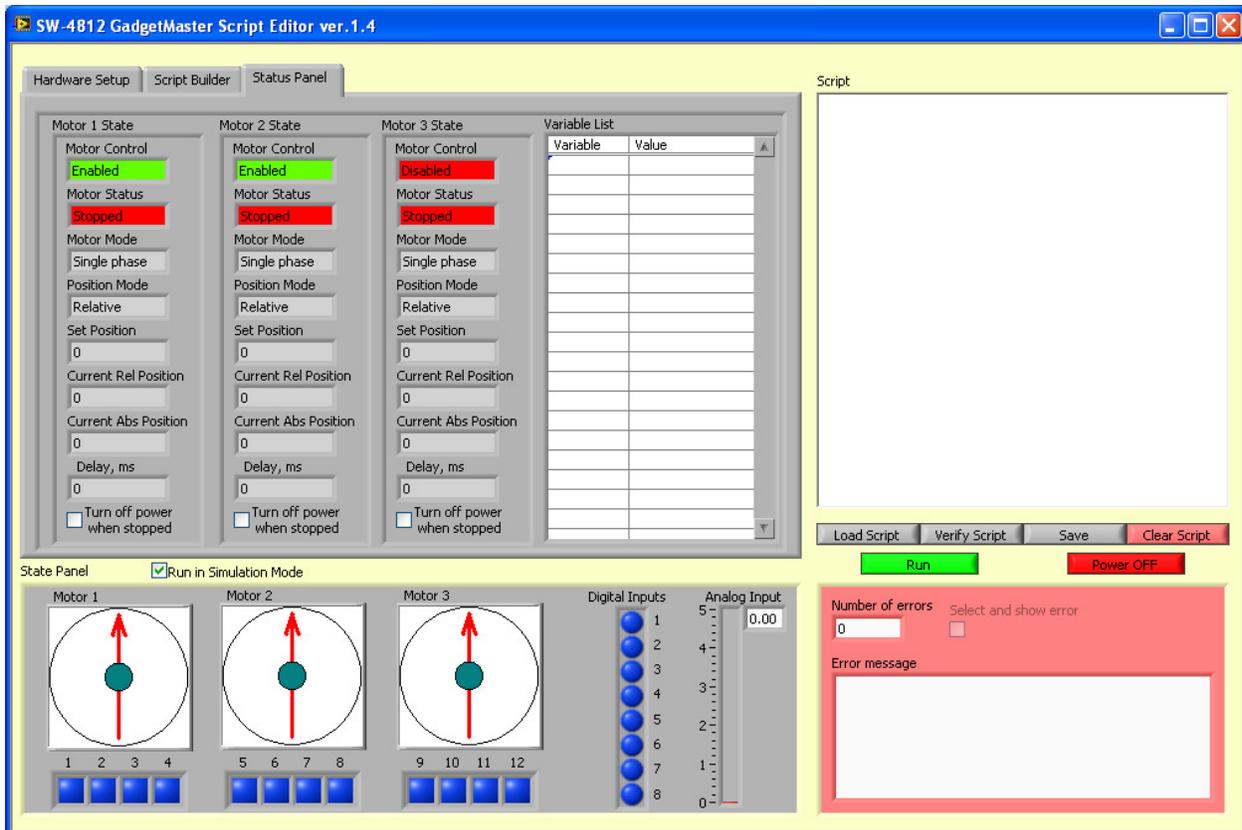
Figure 3 Status Panel

Notes on the Script

1.   The script can be written using capital and small letters; in general GadgetMaster Script Editor does differentiate between capital and small letters.

2.   Spaces in the script are generally not important and even not required in most cases. It is, however, recommended to use them as it increases the readability of the program text.

3.   Operators in the program are should be separated by semicolon ";" or start from a new line. Semicolon sign is not required at the end of a line. The built-in script generator creates a script where all operators start from a new line. That is done to keep the script structure very simple and easy to read. If desired, to shorten programs you can manually edit the script and to combine a few operators separated by semicolon into one line.

4.   An operator may have a text label, which is written before the operator and separated from it by a colon sign ":".

For example
A: Move(1) = 100   "this would move motor 1, 100 steps"

It is allowed to use variables to read various board related parameters such as digital and analog inputs or to write simple mathematical and logical expressions. Each variable name should start with a letter and should not be more than 2 symbols length. The second symbol is optional and it can only be a digit.

For example
X1 = 100
Move(1) = X1   "this would also move motor 1, 100 steps"

5.   It is possible to assign a text string to a variable; the text should be included within double quotation marks:
a="Line"
b="line"

6.   As it is mentioned earlier GadgetMaster Script Editor ver.1.4 does not distinguish between letter cases. Thus variables **a** and **b** are considered to have the same value.

7.   GadgetMaster Script Editor ver.1.4 supports few types of variables: Integer number (I32), Double precisions floating number (DBL), string. It is possible to assign the same variable to different data types (string, integer number, double precision floating number and so on). The last assignment determines the type of the variable.

8.   GadgetMaster Script Editor ver.1.4 does not support array variable type and complex numbers.

9.   It is possible to write comments in the text of the program. A line starting with a simple or double quotation mark is considered as a comment. It is possible to put a comment inside a line after an operator.

For example

' This is a comment

List of Operators

1. Port Address=[address]
    Set parallel port address.
    Address format: First "H" (or "h") letter followed by hexadecimal port address. Maximal allowed port address is FFFF.
    Examples:
    Port Address=H378
    Port Address=H3BC
    Port Address=HFEF0

2. [variable]=Port Address
    Read parallel port address into specified variable. Variable type is I32.
    Examples:
    a=Port Address

3. Jumper=[position]
    Set jumper position.
    Only 2 positions are allowed: **A** and **B.**
    Examples:
    Jumper=A
    Jumper=B

4. [variable]=Jumper
    Read jumper position.
    Example:
    a=Jumper

5. Control(#motor)=[state]
    Set the selected stepper motor into *Enabled* or *Disabled* state.
    Only these states are allowed. This operator overwrites the *Motor Connection* selection made in the *Hardware Setup* tab. #motor is a motor number. Only digits 1, 2 and 3 are allowed.
    Examples:
    Control(1)=Enabled
    Control(2)=Disabled

6. [variable]=Control(#motor)
    Read the motor state.
    Examples:
    a=Control(1)

7. Motor Mode(#motor)=[mode]
    Set the selected stepper motor into *Single phase, Half step* or *Two phases* mode.
    Only these modes are allowed. This operator overwrites the *Motor Mode* selection made in the *Hardware Setup* tab.
    Examples:

Motor Mode(1)=Single phase
Motor Mode(2)=Half step
Motor Mode(3)=Two phases

8. [variable]=Motor Mode(#motor)
Read the motor mode.
Example:
a=Motor Mode(1)

9. Position Mode(#motor)=[position mode]
Set motor position mode into *Relative* or *Absolute*.
Only these modes are allowed. This operator overwrites the *Position Mode* selection
made in the *Hardware Setup* tab.
Examples:
Position Mode(1)=Relative
Position Mode(2)=Absolute

10. [variable]=Position Mode(#motor)
Read position mode.
Example:
a=Position Mode(1)

11. Set Position(#motor)=[value or variable]
Set motor target position.
Value should be a number. It is possible to use a variable that was assigned value before
or an arithmetic expression. Set Position is considered as Relative or Absolute depending
of the Position Mode.
Examples:
Set Position(1)=200
a=150; Set Position(2)=a
Set Position(3)=a*sqrt(2)

12. [variable]=Set Position(#motor)
Read Set Position.
Examples:
a=Set Position(1)

13. [variable]=Rel Position(#motor)
Read current Relative Position.
Current relative position is determined as a number of steps moved by a motor since the
last move command was executed.
Examples:
a=Rel Position(1)

14. [variable]=Abs Position(#motor)
Read current Absolute Position.

Examples:
a=Abs Position(1)

15. Delay(#motor)= [value/variable/expression]
    Set delay time in ms between motor steps.
    This command enables the changing of the motor time delay and can be used while the
    motor is moving. Due to hardware limitation the smallest time delay is 2 ms.
    Examples:
    Delay(1)=10
    a=20; Delay (2)=a
    Delay (3)=a+1

16. [variable]=Delay(#motor)
    Read Delay value.
    Examples:
    a=Delay(1)

17. Stopped Power(#motor)=[on/off]
    Set motor power when motor motion is completed.
    This operator takes effect until the setting is changed by the same operator with different
    argument.
    Examples:
    Stopped Power(1)=on
    Stopped Power(2)=off

18. [variable]=Stopped Power(#motor)
    Read Stopped Power setting.
    Variable type is Boolean. TRUE value means that the power is set to OFF. FAULSE
    value means that the power is set to ON.
    Examples:
    a= Stopped Power (1)

19. [variable]=Motor Status(#motor)
    Read current motor status: "Stopped" or "Moving".
    Example:
    a=Motor Status(1)

20. Move(#motor)[w]
    Move motor to set position.
    Parameter [w] is optional. If there is no [w] parameter the operator starts the motion and
    the next operator is immediately executed. There are some operators that cannot be
    executed when the motor is moving. In this case the program generates an error and will
    be aborted unless the operator "ON ERROR" is active (see below).

    If parameter [w] is present the operator starts the motion and waits until the motion is
    completed. Only after that the next operator is getting executed.

Example:
Move(1)
Move(2)w

21. Move(#motor)[w]= [value/variable]
    Set motor position and starts motion.
    Set Position is considered as Relative or Absolute depending of the Position Mode.
    Parameter [w] is optional and it works the same way as described above for
    Move(#motor) operator.
    Example:
    Move(1)=200
    Move(2)w=200

22. Stop(#motor)
    Stop moving motor.
    An error is generated if a stopped motor receives this command. The error code is 1000.
    Example:
    Stop(1)

23. Set Home(#motor)
    Set current motor position as zero absolute position.
    The motor status should be "Stopped", otherwise an error with error code 5012 is
    generated.
    Example:
    Set Home(1)

24. Power Off(#motor)
    Set motor power off.
    The motor status should be "Stopped", otherwise an error with error code 5015 is
    generated.
    Example:
    Power Off(1)

25. Wait[value/ variable/ expression]
    Causes the program to wait the specified number of milliseconds.
    Example:
    Wait(150)
    Wait (t5)
    Wait(t1*10)

26. Pause
    Causes the program to pause execution.
    A blinking red button appeares in the left bottom corner of the GadgetMaster ver.1.4.
    Pressing this button resumes the execution.
    Example:
    Pause

27. Goto [label]
    Transfers the program execution to the operator with given label.
    Example:
    Goto A

28. On Error [label]
    Sets the program behaviour when an error is generated.
    Parameter [label] is optional. If [label] is present once an error is generated the program
    execution transfers to the operator with given label. This setting takes effect until new
    operator "On Error" is executed.

    If there is no [label] parameter in the " ON Error" operator than once an error is
    generated the program is aborted.
    Example:
    On Error A
    On Error

29. Output(#bit)=[value]
    Sets Digital Output at #bit to the specified value.
    Value can only be 0 or 1. Only one bit can be changed at a time. The values of other bits
    are not affected. The motor control should be disabled otherwise an error with code 5007
    is generated.
    Example:
    Output(1)=0
    Output(2)=1

30. [variable]=Output(#bit)
    Reads the value of the Digital Output at #bit.
    Variable type is I32.
    Example:
    a=Output(1)

31. [variable]=Digital Input(#bit)
    Reads the value of the Digital Input at #bit.
    Variable type is I32.
    Example:
    a=Digital Input(1)

32. [variable]=Analog Input
    Reads the value of the Analog Input.
    Variable type is DBL.
    Example:
    a=Analog Input

33. [variable]=Time

Reads the system time.
Time is in seconds passed since midnight. Variable type is DBL.
Example:
a=Time

34. [variable]=Date
    Reads the system date.
    Variable type is String.
    Example:
    a=Date

35. For [variable]=[initial value] to [final value] step [value]
    Next [variable]
    Standard "For Next" operator.
    If Step value is 1 it can be omitted.

36. If [condition] then [operators]
    Else [operators]
    End If
    Standard "IF Then Else" operator.
    If two numbers of different type (I32 and DBL) are compared they are transformed into
    DBL type.
    Only "equal" (=) or "not equal" (<>) conditions can be used to compare string variables.
    It is possible to compare Jumper, Motor Control, Motor Mode, Position Mode and Motor
    Status values with strings using only "equal" or "not equal" conditions. The case (upper
    or low) and the spaces are not important for this type of comparison.