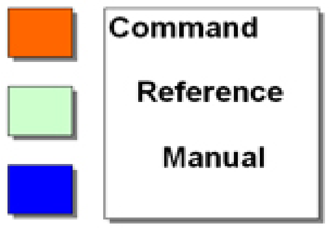
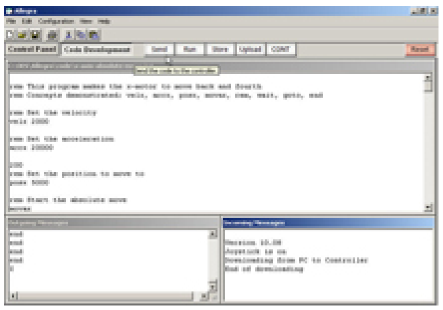
Operating Software Instruction Manual
for
OES Series of Solenoid Control Systems



Command Reference Manual



```
rem This program makes the x-motor to move back and fourth
rem Concepts demonstrated: velx, accx, posx, movax, rem, wait, goto, end

rem Set the velocity
velx 2000

rem Set the acceleration
accx 20000

100
rem Set the position to move to
posx 5000

rem Start the absolute move
movax
```

```
end
end
end
end
0
```

```
Version 10.08
Joystick is on
Downloading from PC to Controller
End of downloading
```

Stand-alone Operation

Host Controlled Operation

Home

GENERAL CONFIGURATION

MSGOFF	Turns off the feedback messages sent from the controller.
MSGON	Turns on the feedback messages sent from the controller.
RSTIMER	Resets the value of the built-in timer to zero.
RTIMER *variable*	Places the value of the built-in timer in in *variable*.

DATA FLOW

CLRBIT *value* or *variable*	Resets the discrete output specified by *value* to low state.
INPUT *variable*	Receives numeric data from the serial port and places it in *variable.*
IN *variable*	Reads the input ports and places its *value* in *variable*.
OUT *value* or *variable*	Writes the *value* or *variable* to the output ports.
PRINT *string, value* or *variable*	Prints the specified *string, value* or *variable* to the serial port.
PWM1 *value* or *variable*	Writes the *value* or *variable* to the PWM1.
PWM2 *value* or *variable*	Writes the *value* or *variable* to the PWM2.
PWM3 *value* or *variable*	Writes the *value* or *variable* to the PWM3.
PWM4 *value* or *variable*	Writes the *value* or *variable* to the PWM4.
SETBIT *value* or *variable*	Sets the discrete output specified by *value* to high state.

PROGRAM FLOW

CONT	Continues the program after a PAUSE command is received.
END	Terminates the program execution.
GOSUB *line*	Branches to subroutine starting at line number specified by *line*.
GOTO *line*	Transfers program execution to line number specified by *line*.
IF *expr* THEN *statement*	If *expr* is true executes *statement* , otherwise continues execution on to next line.
IF *expr* THEN *statement1* ELSE *statement2*	If *expr* is true executes *statement1* , otherwise executes *statement2*.
IFBIT *value* THEN *statement*	If the specified bit is high then executes *statement*.
IFBIT *value* THEN *statement1* ELSE *statement2*	If the specified bit is high then executes *statement1* , otherwise executes *statement2*.
IFNOTBIT *value* THEN *statement*	If the specified bit is low then executes *statement*.
IFNOTBIT *value* THEN *statement1* ELSE *statement2*	If the specified bit is low then executes *statement1* , otherwise executes *statement2*.
REM	Designates a comment line.
RETURN	Ends a subroutine and causes execution to resume at the statement after the line, which called the subroutine.
PAUSE	Pauses program execution until CONT is received.
WAIT *value* or *variable*	Waits for *value* number of milliseconds.

MISCELLANEOUS

DOWNLOAD	Prepares the controller to receive the BASIC code. The mode is terminated upon receiving a dollar sign "$".

LIST	Sends the current program to the serial port.
NEW	Purges the memory.
ROM	Used for updating the controller code
RUN	Runs the program.
SAVE	Saves the current program in the non-volatile memory.
UPLOAD	Retrieves the program from the non-volatile memory into controller’s RAM.

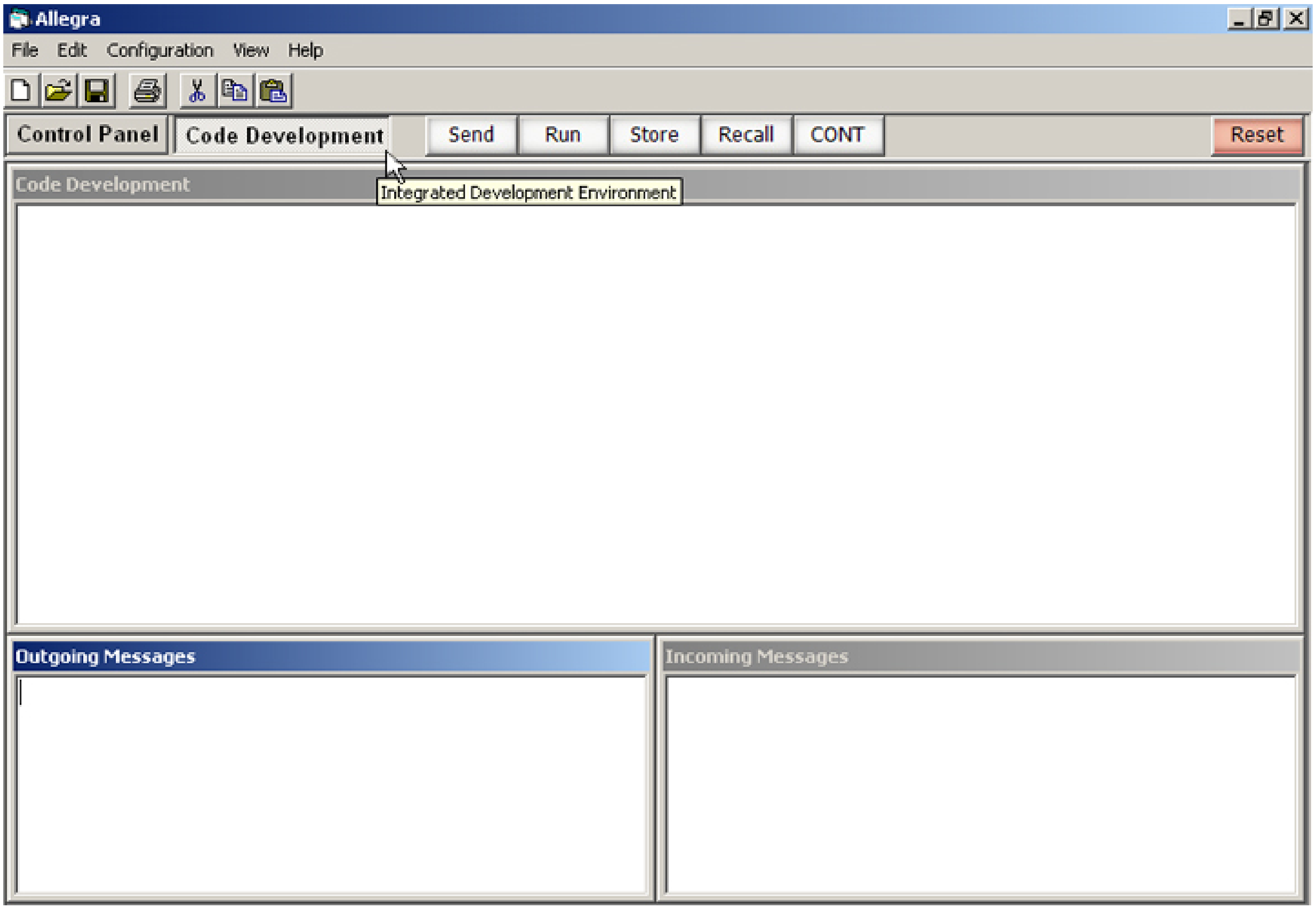
VARIABLES and OPERATIONS	
Logical and Mathematical	
RAND *variable*	Assigns a random value to *variable*

Home

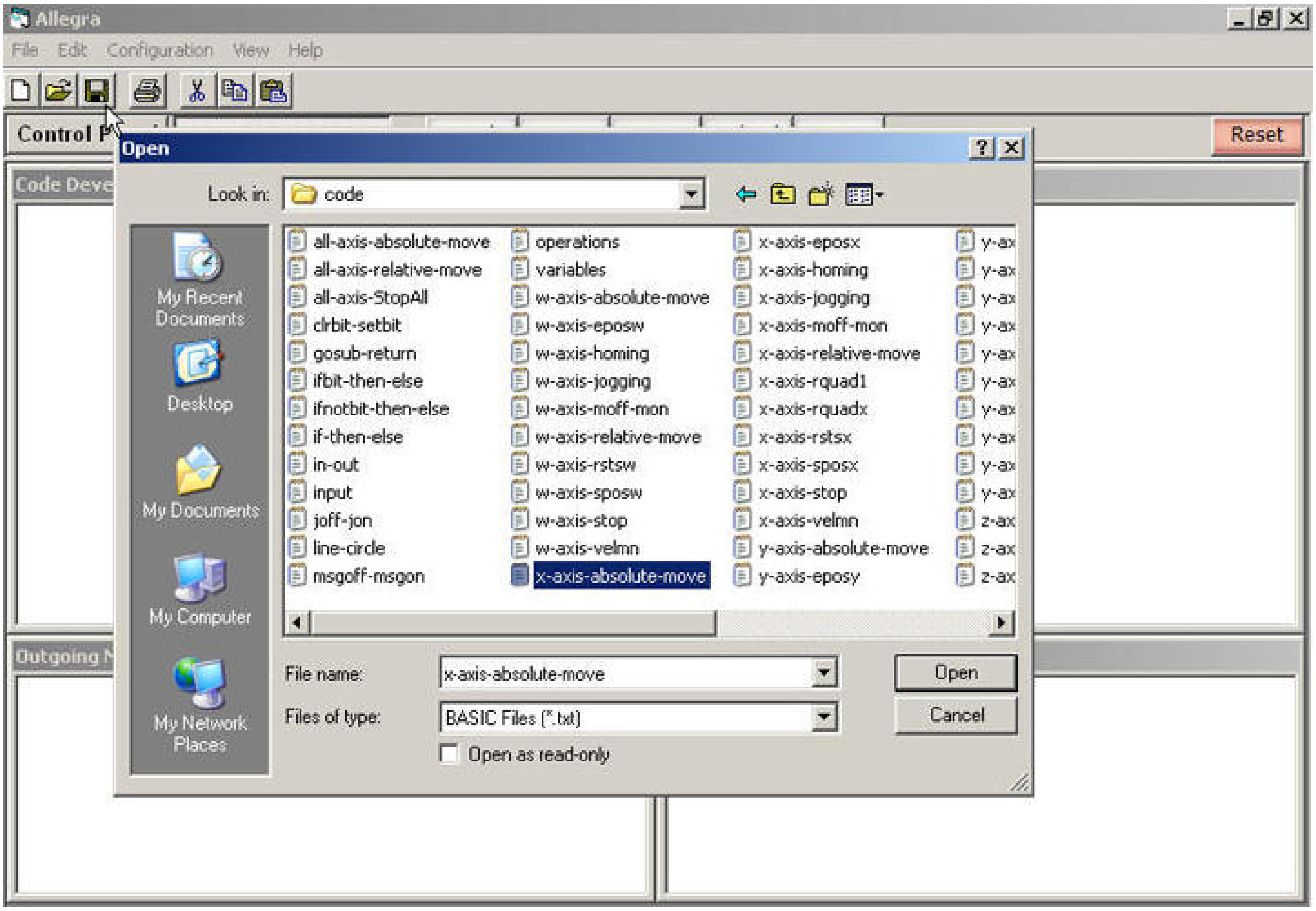
Stand-alone Operation

The following are the steps that need to be taken to write, download and run the code.

1) Click on 'Code Development'.



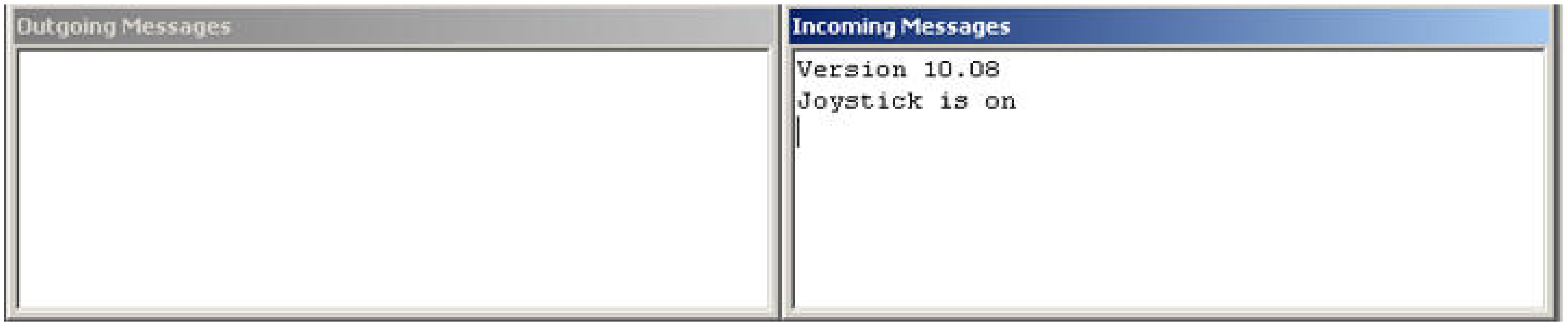
2) Click on ‘Open File’ icon to open an existing file in the default directory; for example, pwm.txt.



File name: x-axis-absolute-move

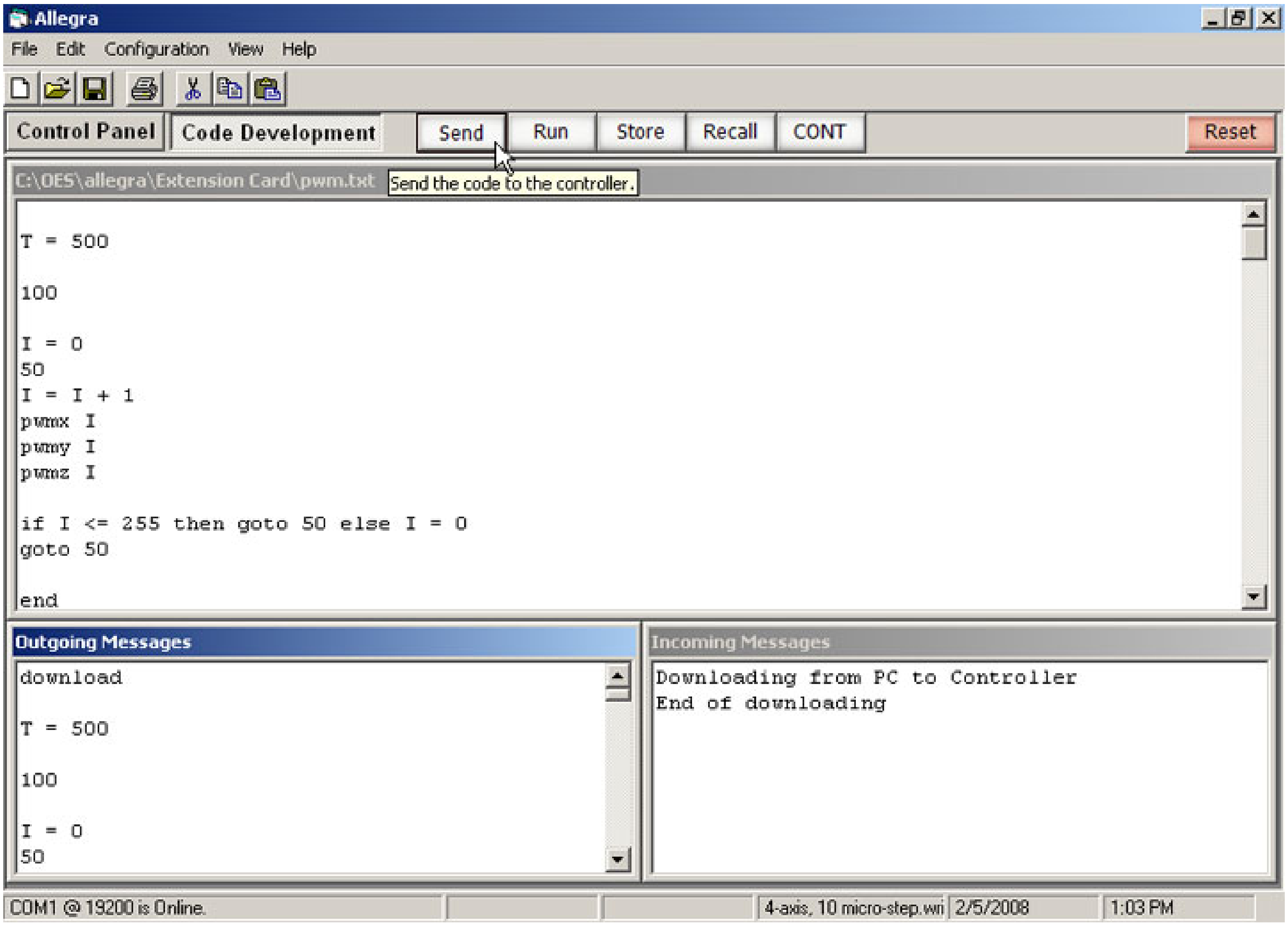
Files of type: BASIC Files (*.txt)

3) Click on the Reset button before each download. Please make sure the status LED blinks then remains lit. The revision number should be displayed in the Incoming Messages pane.



```
Version 10.08
Joystick is on
```

4) Click on Send button to send the code from PC to the controller.


Allegra
File Edit Configuration View Help
Control Panel
Code Development
Send
Run
Store
Recall
CONT
Reset
C:\OES\allegra\Extension Card\pwm.txt
Send the code to the controller.
Outgoing Messages
Incoming Messages

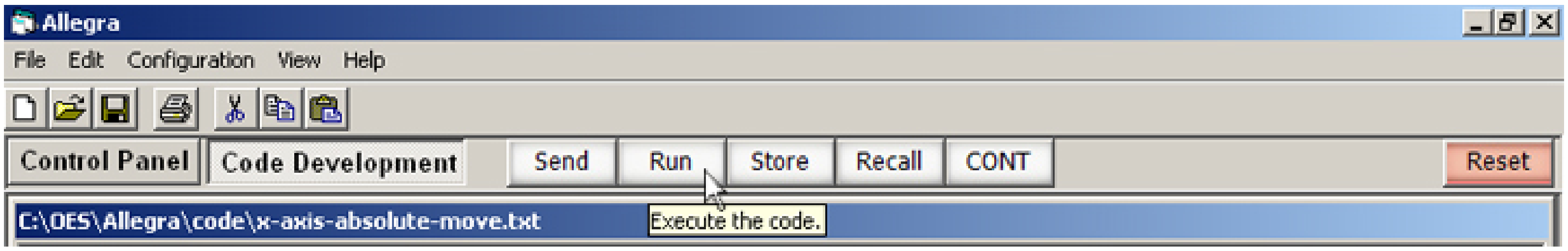
```
T = 500

100

I = 0
50
I = I + 1
pwmx I
pwmy I
pwmz I

if I <= 255 then goto 50 else I = 0
goto 50

end
```

```
download

T = 500

100

I = 0
50
```

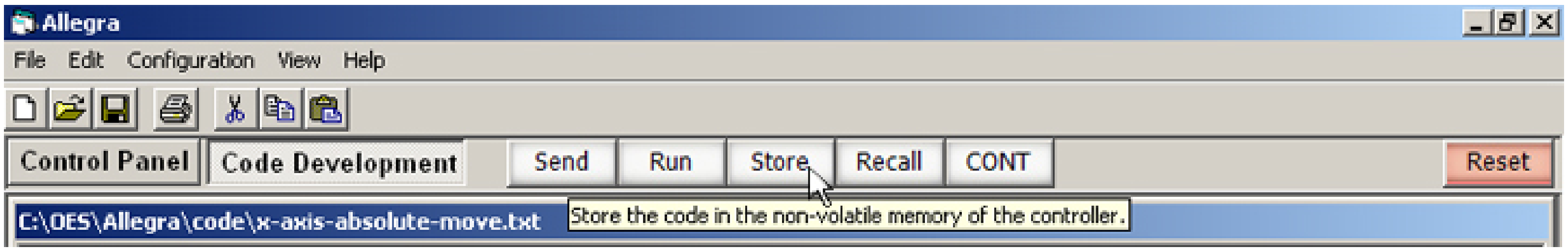
```
Downloading from PC to Controller
End of downloading
```

At the start of download the controller sends "Downloading from PC to Controller" which is displayed in the Incoming Messages. When the download is completed the controller sends "End of downloading" message to the PC.

5) Once the download is completed, click on Run and the controller will start to execute the program.



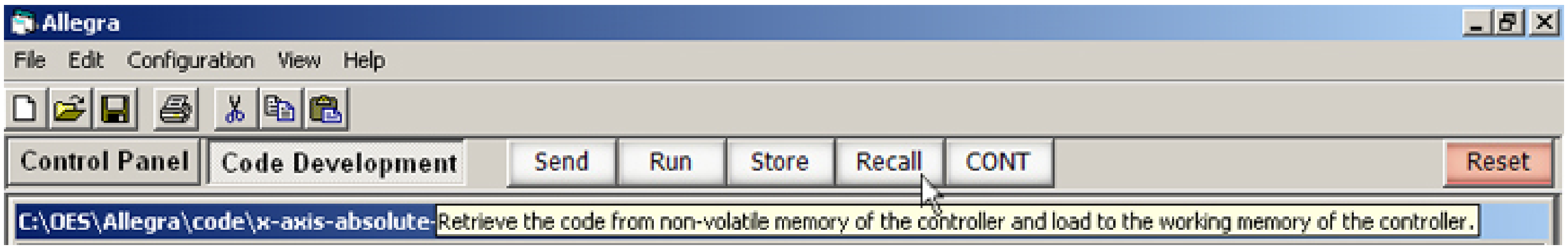
6) To store the code to non-volatile memory of the controller, repeat step 4 and 5 then click on Store. This will save the program in the non-volatile memory of the controller.



C:\OES\Allegra\code\x-axis-absolute-move.txt

Store the code in the non-volatile memory of the controller.

7) To retrieve the saved code from the non-volatile memory and run the code , click on Recall then Run.



C:\OES\Allegra\code\x-axis-absolute-

Retrieve the code from non-volatile memory of the controller and load to the working memory of the controller.

Or, you could use the Upload-and-Run pin of the command port. Please consult the Hardware Reference Manual for the location of the pin.

Host Controlled Operation

In this mode the host will send a series of ASCII commands to the controller over the RS-232 serial port. The controller process to the incoming commands and responses with the proper messages.

Programming Example in Visual BASIC

The following example turns on the outputs 1 and 2.

```
Private Sub Command1_Click()

        'Function Prototype
        Declare Function SioPuts Lib "WSC32.DLL" (ByVal Port As Long, ByVal Buffer As String,
        ByVal Size As Long) As Long

        Dim Code As Long
        Dim StringToBeTransmtd As String
        ' turn on output 1
        StringToBeTransmtd = "setbit 1" + vbCr
        Code = SioPuts(ThePort, StringToBeTransmtd, Len(StringToBeTransmtd))

        ' turn on output 2
        StringToBeTransmtd = "setbit 2" + vbCr
        Code = SioPuts(ThePort, StringToBeTransmtd, Len(StringToBeTransmtd))

End Sub
```

Programming Example in 'C'

The following example turns on the outputs 1 and 2.

```
void send_command(void)
{

        char StringToBeTransmtd[80];

        //  turn on output 1
        strcpy(StringToBeTransmtd,"setbit 1\n");
        SioPuts(Port,StringToBeTransmtd,strlen(StringToBeTransmtd));

        // turn on output 2
        strcpy(StringToBeTransmtd,"setbit 2\n");
        SioPuts(Port,StringToBeTransmtd,strlen(StringToBeTransmtd));

}
```

The following is the information that you need to establish communication with OES line of controllers.

1) The baud rate is 19.2 K, 8-bit, no parity, one stop bit.

2) The Request-to-Send (RTS) of the serial port is used to reset the controller card. During initialization

you would have to set this line to Clear. There is a jumper on the controller card that disables this function (You won't be able to hard reset the controller.)

3) To reset the controller, set the line to Set wait for at least 10 msec, then set it back to Clear.

4) The length of input buffer of the OES' controller is 256 bytes.

5) ASCII characters should be terminated with CR or LF.

6) After sending each packet of data to the OES' controller, sufficient time should be given to the controller to process it, usually 100msec.

To receive characters, a buffer is setup and all the incoming characters are stored in it until they are fetched by the application.

OES uses a serial communication package from www.marshallsoft.com.

Copyright © 2002, 2003, 2004, 2005, 2006, 2007, 2008 Optimal Engineering Systems, Inc.

msgoff

Syntax
msgoff

Function
Turns off the feedback messages sent from the controller.

Controller Returns
None

Mode
Run, Command

Example
msgoff

msgon

Syntax
msgon

Function
Turns on the feedback messages sent from the controller.

Controller Returns
None

Mode
Run, Command

Example
msgon

rstimer

Syntax
rstimer

Function
Resets the value of the built-in timer to zero.

Range of Value
None

Controller Returns
None

Mode
Run, Command

Example
rstimer

rtimer

Syntax
rtimer *variable*

Function
Places the value of the built-in timer in in *variable*. The unit is milli-seconds. If no *variable* is specified the controller sends the value to the serial port.

Range of Value or Variable
0 through +2147483647

Controller Returns
The value if no argument is specified.

Mode
Run, Command

Example
```
rtimer A
print "value of timer = ", A
```

Back Home

clrbit

Syntax

clrbit *value*
clrbit *variable*

Function

Resets the discrete output specified by *value* or *variable*. Refer to the hardware reference manual for the location of each pin.

Range of Value or Variable

1 to 8

Controller Returns

None

Mode

Run, Command

Example

```
rem Reset discrete output 5
clrbit 5
A = 7
rem Reset discrete output 10
clrbit A
```

setbit

Syntax

setbit *value*
setbit *variable*

Function

Sets the discrete output specified by *value* or *variable*. Refer to hardware reference manual for the location of each pin.

Range of Value or Variable

1 to 8

Controller Returns

None

Mode

Run, Command

Example

```
setbit 5
A = 7
setbit A
```

Back Home

bfr

Syntax
bfr *variable*

Function
Places the number of the characters to be transmitted in *variable*. If no *variable* is specified the controller sends the value to the serial port.

Range of Value
0 through 255

Controller Returns
The value if no argument is specified.

Mode
Run, Command

Example
bfr A
print "The number of characters to be transmitted = ”, A

chr

Syntax
chr *value* or *variable*

Function
Sends a single byte character specified by value or variable to the serial port.

Range of Value
0 through 255

Controller Returns
None

Mode
Run, Command

Example
chr 100

input

Syntax
input *variable*

Function
Receives numeric data from the serial port and places it in *variable*.

Range of Variable
-2147483647 to +2147483647

Controller Returns
“?”

Mode
Run, Command

Example
input A

rem A numeric value should be sent to the controller through the serial port

print

Syntax
print *value*
print “*string*”
print “*string*”, *value*

Function
Prints the specified *value* or *string* to the serial port.

Range of Value
-2147483647 to +2147483647

Controller Returns
string, value

Mode
Run, Command

Example

```
A = 123
print “A =”, A
```

send

Syntax

send *value*
send “*string*”
send “*string*”, *value*

Function

Sends the specified *value* or *string* to the serial port without carriage return or line feed characters.

Range of Value

-2147483647 to +2147483647

Controller Returns

string, value

Mode

Run, Command

Example

```
A = 123
send  “A =”, A
```

in

Syntax
in *variable*

Function
Reads a word from the input ports and places it in *variable*. Refer to the hardware reference manual for the location of each pin.

Range of Value
0 to 65535

Controller Returns
None

Mode
Run, Command

Example
in A

out

Syntax
out *value*
out *variable*

Function
Writes the *value* or *variable* to the output ports. Refer to the hardware reference manual for the location of each pin.

Range of Value
0 to 255

Controller Returns
None

Mode
Run, Command

Example

```
A = 10
out A
```

Back Home

pwm1

Syntax
pwm1 *value*
pwm1 *variable*

Function
Writes the *value* or *variable* to the PWM1.

Range of Value or Variable
0 to 127

Controller Returns
None

Mode
Run, Command

Example
pwm1 123
A = 32
pwm1 A

pwm2

Syntax
pwm2 *value*
pwm2 *variable*

Function
Writes the *value* or *variable* to the PWM2.

Range of Value or Variable
0 to 127

Controller Returns
None

Mode
Run, Command

Example
pwm2 123
A = 32
pwm2 A

pwm3

Syntax
pwm3 *value*
pwm3 *variable*

Function
Writes the *value* or *variable* to the PWM3.

Range of Value or Variable
0 to 127

Controller Returns
None

Mode
Run, Command

Example
pwm3 123
A = 32
pwm3 A

pwm4

Syntax
pwm4 *value*
pwm4 *variable*

Function
Writes the *value* or *variable* to the PWM4.

Range of Value or Variable
0 to 127

Controller Returns
None

Mode

Run, Command

Example

```
pwm4 123
A = 32
pwm4 A
```

cont

Syntax
cont

Function
Continues the execution of the program after a pause command is received.

Controller Returns
None

Mode
Run

Example
rem Halt program execution. The controller may be interrogated after receiving the pause.
pause
rem Continue the program execution.
cont

pause

Syntax
pause

Function
Pauses program execution until CONT is received. PAUSE is useful in debugging. Using PAUSE, the execution of the program may be halted and the controller can be interrogated.

Range of Value
None

Controller Returns
Pausing...Type CONT.

Mode
Run

Example
if A = B then pause

end

Syntax

end

Function

Ends program execution, and returns to command level.

Controller Returns

Ending

Mode

Run

Example

```
rem Assign a value to A
A = 1
rem Print the value stored in A
print A
rem End program execution
end
```

gosub

Syntax
gosub *line*

Function
Branches to subroutine starting at line number specified by *line*.

Controller Returns
None

Mode
Run

Example
```
gosub 100
rem other commands
…
end
100 rem commands
…
return
```

return

Syntax
return

Function
Ends a subroutine and causes execution to resume at the statement after the line that called the subroutine.

Controller Returns
None

Mode
Run

Example
```
gosub 100
rem other commands
```

```
…
end
100 rem Subroutine starts here
…
return
```

goto

Syntax
goto *line*

Function
Transfers program execution to line number specified by *line*.

Controller Returns
None

Mode
Run

Example
```
goto 100
rem other commands
…
end
100 rem commands
…
return
```

if…then

Syntax
if {relational expression} then *instruction*

Function
If relational expression is true execute *instruction.*

Error Message
If then is missing, “Expected: then”

Mode
Run

Example
if A<> B then print “A<>B”

if…then…else

Syntax
if {relational expression} then *instruction1* else *instruction2*

Function
If relational expression is true execute *instruction1*, otherwise execute *instruction2.*

Error Message
If then is missing, “Expected: then”

Mode
Run

Example
if A<> B then print “A<>B” else print “A=B”

Back Home

ifbit…then

Syntax
ifbit *value* then *instruction*
ifbit *variable* then *instruction*

Function
If the bit specified by *value* or *variable* is high then execute *instruction*. Refer to the hardware reference manual for the location of each pin.

Range of Value or Variable
1 to 16

Controller Returns
None

Mode
Run

Example
A = 10

ifbit A then goto 100

ifbit…then…else

Syntax
ifbit *value* then *instruction1* else *instruction2*
ifbit *variable* then *instruction1* else *instruction2*

Function
If the bit specified by *value* or *variable* is high then execute *instruction1*, otherwise execute *instruction2*. Refer to the hardware reference manual for the location of each pin.

Range of Value or Variable
1 to 16

Controller Returns
None

Mode

Run

Example

```
A = 10
ifbit A then goto 100 else goto 200
```

ifnotbit…then

Syntax
ifnotbit *value* then *instruction*
ifnotbit *variable* then *instruction*

Function
If the bit specified by *value* or *variable* is low then execute *instruction*. Refer to the hardware reference manual for the location of each pin.

Range of Value or Variable
1 to 16

Controller Returns
None

Mode
Run

Example
A = 10
ifnotbit A then goto 100

ifnotbit…then…else

Syntax
ifnotbit *value* then *instruction1* else *instruction2*
ifnotbit *variable* then *instruction1* else *instruction2*

Function
If the bit specified by *value* or *variable* is low then execute *instruction1*, otherwise execute *instruction2*. Refer to the hardware reference manual for the location of each pin.

Range of Value or Variable
1 to 16

Controller Returns
None

Mode

Run

Example

```
A = 10
ifnotbit A then goto 100 else goto 200
```

rem

Syntax
rem

Function
Designates a comment line.

Controller Returns
None

Mode
Run

Example
rem This is a comment line.

wait

Syntax
wait *value*
wait *variable*

Function
Waits for *value* or *variable* number of milliseconds.

Range of Value or Variable
0 – 65535

Controller Returns
None

Mode
Run

Example
```
 rem Wait for 10 mSecs
wait 10
rem Wait for 200 mSecs
A = 200
wait A
```

downlaod

Syntax
download

Function
Signals the controller to receive the BASIC code. The mode is terminated upon receiving a dollar sign “$”.

Controller Returns
Downloading from PC to Controller
(After completion of downloading)
End of downloading

Mode
Command

Example
rem Signal the controller to receive the BASIC code. The code should be terminated by a dollar sign “$”.
download

list

Syntax
list

Function
Sends the current program from the controller to the serial port.

Controller Returns
Listing

Mode
Command

Example
list

new

Syntax
new

Function
Purges the controller’s memory.

Controller Returns
Purging the memory

Mode
Command

Example
new

rom

Syntax
rom

Function
Used for updating the controller's firmware code.

Range of Value
None

Controller Returns
ROMing

Mode
Command

Example

run

Syntax
run

Function
Runs the program.

Controller Returns
Running

Mode
Command

Example
run

save

Syntax
save

Function
Saves the current program in the non-volatile memory of the controller.

Controller Returns
Wrote BASIC to non-volatile.
Verified BASIC

Mode
Command

Example
save

upload

Syntax
upload

Function
Retrieves the program from the non-volatile memory of the controller.

Controller Returns
BASIC code uploaded

Mode
Command

Example
```
upload
```

variables

The controller recognizes twenty two 32-bit variables, A through V. These variables may be used to receive, store and process numeric values. The range of the variables is from -2147483647 through +2147483647.

logical and mathematical operations

The following operation may be performed on any two values or variables.

+	Addition
/	Division
<	Less Than
<=	Less Than or Equal
&	Logical AND
|	Logical OR
#	Logical XOR
%	Modulo
>	More Than
>=	More Than or Equal
*	Multiplication
<>	Not Equal
~	Logical One's Complement
()	Parentheses
-	Subtraction

rand

Syntax
rand *variable*

Function
Places a 32-bit random value in the *variable*.

Range of Value or Variable
0 - 4294967295

Controller Returns
None

Mode
Run, Command

Example

```
rem a random value will be placed in B
rand B
print B
```