CONTROL TECHNOLOGY CORPORATION

Model 5200 Script Language Guide

Model 5200 Script Language Guide

Blank

 **WARNING:** Use of CTC Controllers and software is to be done only by experienced and qualified personnel who are responsible for the application and use of control equipment like the CTC controllers. These individuals must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and/or standards. The information in this document is given as a general guide and all examples are for illustrative purposes only and are not intended for use in the actual application of CTC product. CTC products are not designed, sold, or marketed for use in any particular application or installation; this responsibility resides solely with the user. CTC does not assume any responsibility or liability, intellectual or otherwise for the use of CTC products.

The information in this document is current as of the following Hardware and Firmware revision levels. Some features may not be supported in earlier revisions. See www.ctc-control.com for the availability of firmware updates or contact CTC Technical Support.

Model Number	Hardware Revision	Firmware Revision
5200	All Revisions	>= 5.00.31

TABLE OF CONTENTS

CHAPTER

1

Introduction

As an enhancement to Quickstep, a Script Language is available added that allows you to do many administrative tasks from within the language. The 5200 Controller can perform numerous operations from within this language, either under program control, and/or via an FTP/Telnet command line interface. These commands can be grouped into a file and executed in batch mode or individually. The entire WebMON administrative and diagnostic environment interfaces with the 5200 by the use of interactive scripting, thus its powerful capabilities. Some of the functionality available is:

Under Quickstep Program Control:

1. Save sets of registers and restore from flash disk. Saving dynamically creates a new script file for later playback.
2. Load alternate Quickstep programs for execution.
3. Run/load 'C' programs.
4. Ability to run script command files (`.ini` files). Nested scripts are also allowed. Script commands can contain any commands.

FTP/Telnet:

1. Retrieve firmware version information on all processors in the system and installed modules.
2. Update of firmware for all processors and modules.
3. Control of Quickstep program state, RESET, RUNNING, RESTART…
4. Query of Quickstep program state.
5. Ability to read/write registers and blocks of registers.
6. Ability to run script command files (`.ini` files).
7. Load alternate Quickstep programs for execution.
8. Administer flash disks, virtual directory mapping, formatting, disk commands.
9. Advanced remote diagnostics.

Script commands may be run both from a text file and/or interactively from the command line of FTP and Telnet. Advanced scripting is discussed in the *Model 5200 Logging and FTP Client Applications Guide, 951-520015*.

Script File Naming/Access

Script files are text files consisting of various commands as defined in Chapter 2. All script files must reside within the flash disk directory `/_system/Scripts` and are of the naming convention `Script001.ini`, `Script002.ini`, … The 001, 002 … numeric part of the file name references the value that must be written to the ***Script Register*** (12311) in order for that script to be executed. For example for a Quickstep program to run `Script001.ini` it would write a 1 to the ***Script Register***. That task would then run the Script to completion before any other task was allowed to execute. Upon execution a read of the Script Register will return the number of the last script executed. A read of the ***Script Result Register*** (12312) will return 0 if busy executing, a 1 if successful and complete, otherwise an error code.

Boot Script

At power up and system reset a script file by the name of `_startup.ini` will automatically be executed if it is resident on the flash disk within the `/_system/Scripts` sub-directory. All script commands are valid within this initialization file, in addition to special unique system configuration groups detailed in Chapter 4. A reset will be required to activate any IP address informational changes, i.e., IP, subnet mask, or gateway. Refer to Chapter 4 for additional information on network parameters. Be aware that if you change any IP information using CTCMON and the `_startup.ini` file also modifies that information, at the next power cycle the executed script will override it. Should you wish to remove the `_startup.ini` file without using FTP or Telnet, you may write a 1 to register 20097 and it will be automatically deleted. This allows you to delete the file should it erroneously disable all connections due to security settings, or should some other setting restrict access. Serial port access is always allowed.

Script Execution

Scripts are executed either via an FTP/Telnet session command line, from within another script, or by writing a value to the ***Script Register***. When executed by a Quickstep program, by writing to the ***Script Register***, the script is run to completion, atomic to (completely within) that 'step'. When executed by an FTP/Telnet session it is run totally in the background and in parallel to Quickstep execution. If issuing critical commands it is recommended that an 'enable safe' command be executed first. This command stops the execution of Quickstep. It is actually good practice to have an 'enable safe' command in the beginning of the script file whenever FTP or external communication may cause a script to execute. This command guarantees no Quickstep task will be executing when the script is executing. 'enable safe' has no effect if executed by a Quickstep program triggered task, since it is atomic to that task. This command is automatically done during firmware updates.

CHAPTER 2

Script Language Commands

This Chapter details all of the various commands available within the 5200 Script Language. Some commands are marked as "Reserved for WebMON", which means they are for reference only due to their complexity and not directly supported when executed at the command line of a FTP or Telnet session

The screen examples shown for the commands described below are from a PC Telnet session. Most may also be run from within FTP using the `ls` command with the command surrounded by double quotes and an `!` preceding the command. For example, `disable debug` becomes `ls "!disable debug"`. Issuing commands other than "!format disk", from within FTP, although work, may cause a connection loss since the FTP protocol is not designed for the amount of information that may be returned. Typically commands are issued either interactively via Telnet, using WebMON, or from within a script file.

Command	*Description*
Register/Data Table Commands	
#####	Reads the specified register, just as the get register command does. To read a block of registers use #####-#####. For example, to read 1000 to 1050, enter 1000-1050. You may also list other registers, using a comma delimiter, in the following format: #####-#####, #####-#####, ##### …
get datatable <row>,<col>	Return the requested <row> and <column> position in numeric format.

get datatable size	Returns the current size, rows/columns, of the data table.
	`BlueFusion/>get datatable size`
	`Rows = 65535, Columns = 0`
	`BlueFusion/>`
	A column size of 0 means the data table is empty.

get datatable row #	Returns the specified data table row data, beginning with '1', as numeric values. Up to 256 bytes will be returned per line. If no row exists for that requested the

Command	Description

	following message will be returned:
	`BlueFusion/>get datatable 1`
	`ERROR: Invalid column!`
	`BlueFusion/>_`

get register #####
or
#####
 	
get register ##### > file.scr
	Displays the current value of a data register during an FTP/Telnet session. When this command is run within a script, the contents of the register can be written to another script for later playback, if desired. To read a block, use the syntax #####-#####. You may also list other registers, using a comma delimiter (e.g., #####-#####, #####-#####, ##### …). Note that a ? is returned as a value if not a valid register.
	`BlueFusion/>get register 13002`
	`13002 = 25072498`
	
	`BlueFusion/>get register 13002-13005`
	`13002 = 25081762`
	`13003 = 40201`
	`13004 = 1`
	`13005 = 0`
	
	`BlueFusion/>get register 13002,13005,13006`
	`13002 = 25094457`
	`13005 = 0`
	`13006 = ?`
	
	`BlueFusion/>_`
	
	get register 13002 > file.scr
	This would write '13002 = 25072498' to the file file.scr such that if played back via the Script Register a write would occur, initializing 13002 with the value specified.

get resource	See 'get register'
set datatable <row>,<col> = <value>	Set the requested <row> and <column> position to the <value> specifed, in numeric format.

set register ##### = #####	Sets a register to a particular value. A block may be set to a particular value by using #####-##### as a range.
	`BlueFusion/>set register 13002=5, 30 = 6, 35-38=9`
	`13002 = 5`
	`30 = 6`
	`35 = 9`
	`36 = 9`
	`37 = 9`
	`38 = 9`
	
	`BlueFusion/>_`

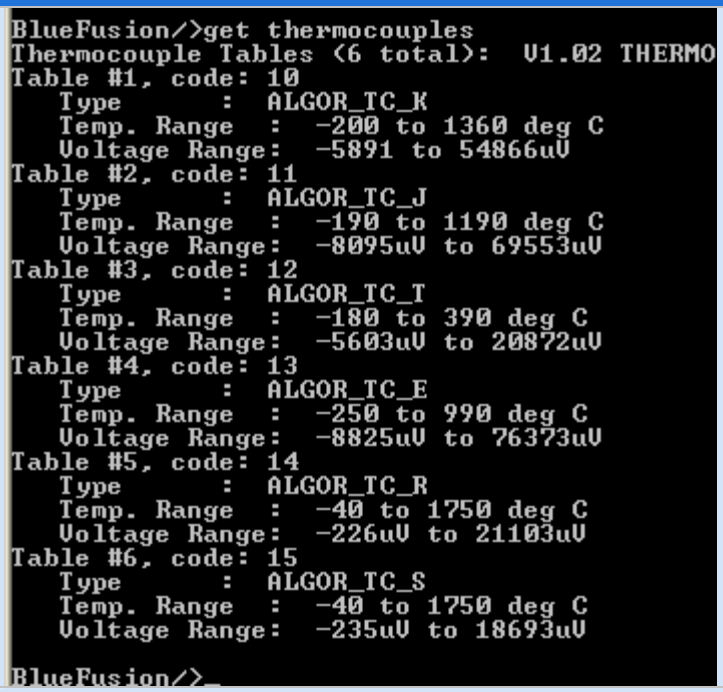
##### = #####	Writes a value to the specified register, just as the set register command does. Note this is the output format used by the run script Script###.ini > Script###.ini command for playback. To clear a block, enter the same syntax as with a read. For example, to clear 1000 to 1050 to 0, enter 1000-1050 = 0.
set resource	See 'set register'.

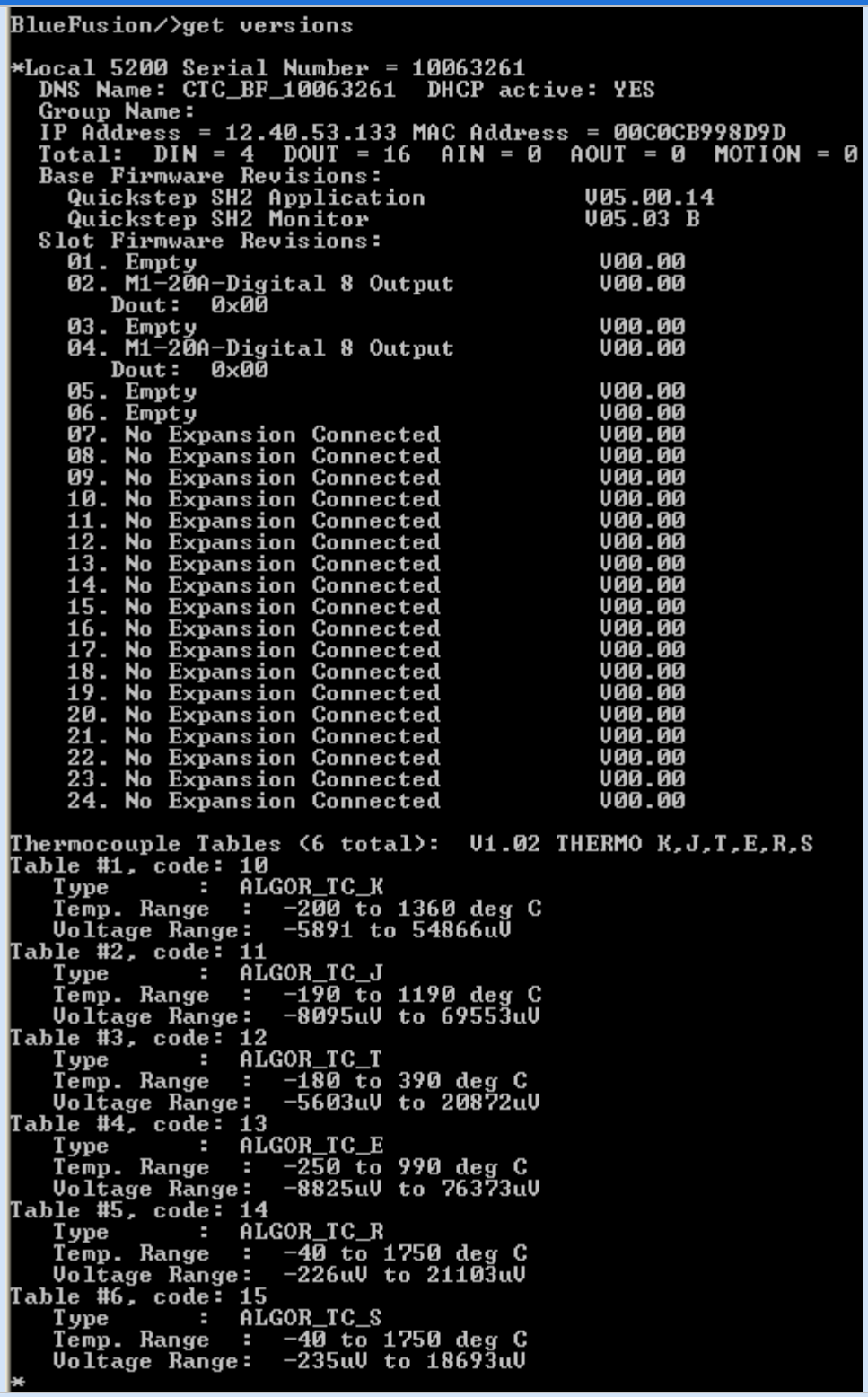
Command	Description
System Commands	

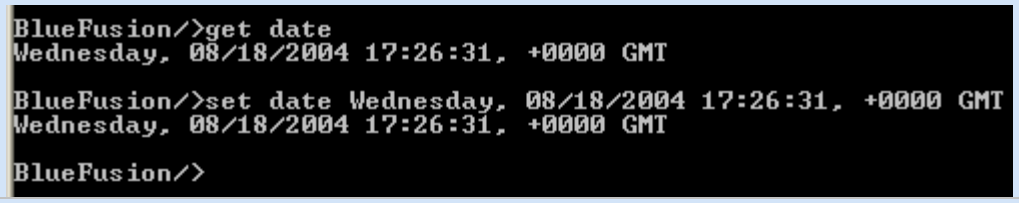
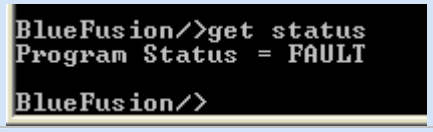
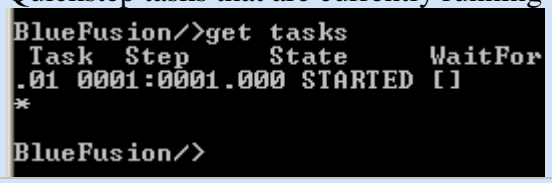
get date
or
date
	Displays the settings of the real-time clock.
	`BlueFusion/>date`
	`Monday, 08/16/2004 15:31:29, +0000 GMT`
	`BlueFusion/>_`

get modules	Returns information about what modules are located in what controller slot locations and their firmware revision level, if relevant. The returned information includes any active 5200 expansion units.
	`BlueFusion/>get modules`
	`01. Empty V00.00`
	`02. M1-20A-Digital 8 Output V00.00`
	`Dout: 0x00`
	`03. Empty V00.00`
	`04. M1-20A-Digital 8 Output V00.00`
	`Dout: 0x00`
	`05. Empty V00.00`
	`06. Empty V00.00`
	`07. No Expansion Connected V00.00`
	`08. No Expansion Connected V00.00`
	`09. No Expansion Connected V00.00`
	`10. No Expansion Connected V00.00`
	`11. No Expansion Connected V00.00`
	`12. No Expansion Connected V00.00`
	`13. No Expansion Connected V00.00`
	`14. No Expansion Connected V00.00`
	`15. No Expansion Connected V00.00`
	`16. No Expansion Connected V00.00`
	`17. No Expansion Connected V00.00`
	`18. No Expansion Connected V00.00`
	`19. No Expansion Connected V00.00`
	`20. No Expansion Connected V00.00`
	`21. No Expansion Connected V00.00`
	`22. No Expansion Connected V00.00`
	`23. No Expansion Connected V00.00`
	`24. No Expansion Connected V00.00`
	`BlueFusion/>_`

get thermocouples	Displays information with regards to what type of thermocouples can be supported using the existing Thermocouples.tbl file in the /_system/Datatables flash disk subdirectory. If this file does not exist, then no thermocouple conversion capability is allowed by the Analog modules.
	If no thermocouple table present:
	`BlueFusion/>get thermocouples`
	`No Thermocouples.tbl file found.`
	`BlueFusion/>_`
	
	If thermocouple table is present:

Command	Description
	
get versions	Displays the current firmware revision levels, serial number, MAC Address of the unit, a listing of the six module bays and the modules installed in them, and supported thermocouples, if any.

Command	Description
	
set date	Set the Real Time clock to the current date/time. The setting of the date is in the exact same format as that returned by the ‘get date’ command. The last GMT parameter is optional:

Command	Description
	
set factory defaults	Restores the non-volatile EEPROM memory to factory settings. This updates items such as serial port baud rate, flash disk size, etc.
set password	Sets the current 'admin' user id password for Telnet and ftp access. A write of a 1 to register 20096 is required to make this password nonvolatile.
Program Control Commands	
attach task	Reserved for WebMON. This command lists the currently attached task. 
attach task ##	Reserved for WebMON. This command used by WebMON to attach itself to a task so that breakpoint information, specific to that task, can be set.
clear all breakpoints	Reserved for WebMON.
clear breakpoints	Reserved for WebMON.
clear breakpoint #	Reserved for WebMON.
continue	Reserved for WebMON. Continue executing a Quickstep task that is stopped due to a breakpoint or is in single step mode.
continue task #	Reserved for WebMON. Continue executing the designated Quickstep task that was stopped due to a breakpoint or is in single step mode.
get status	Displays the current program status: 
get tasks	Reserved for WebMON. This command is used by WebMON to get a list of all Quickstep tasks that are currently running on the controller. 
reset	This command will cause a watchdog reset to occur, resetting the unit as is done during a power up sequence. Any network connections will be lost.
restart task #	Reserved for WebMON. Restart a task at step 1.
run script Script###.ini	Executes the named script. The script file must be resident on the flash disk in the /_system/Scripts subdirectory

Command	Description
run script Script###.ini > Script###.ini	Executes the named script; the contents of any registers that are read, are written to the output named script in such a way that if the output named script file is executed, the register values will be written back to their corresponding registers. This command would cause a get register 13002 command to have a 13002 line written to the designated .ini file. See the ##### (register read) and ##### = ##### (register write) commands, which are alternatives to register get/set commands.
run program <filename.dso>	Loads the named Quickstep program for execution from the flash disk directory /_system/Programs and begins its execution, replacing the present program in memory. This is identical to transferring a program over the serial port or via FTP to non-volatile memory. The program will stay in battery-backed memory until it too, is replaced. Example: run program qsprogram.dso
run userprogram <filename.sr1>	Loads the named 'C' User program for execution from the flash disk directory /_system/Programs and begins its execution, replacing the present program in memory. The 'C' Program will be initialized and run in parallel to any loaded Quickstep programs. Example: run userprogram aload.sr1
set breakpoint #	Reserved for WebMON.
set reset	Sets the current Quickstep program state to RESET.
set restart	Restarts the current Quickstep program.
set running	Resumes the execution of Quickstep programs, from a stopped state.
set stop	Sets a Quickstep program to the stopped state.
step	Reserved for WebMON. Single step a Quickstep task.
step task #	Reserved for WebMON. Single step the designated Quickstep task.
update <filename>	This command is used to re-flash the controller with firmware contained within the /_system/Firmware subdirectory. This would include various modules and the resident communications controller. The exact filename must be specified. Reference the 5200 Remote Administration Guide for details of filename conventions. Example: update com5200V0109.bin
Communications (Networking/Serial) Commands	
comm test	When a loopback cable is connected between COM1 and COM2 this command will test the TX and RX ability of both ports.
disable serial programming	Disables Quickstep programs from being downloaded to the controller via the serial port.
disable sntp monitor	Disables UDP debugger printouts of SNTP packet information. By default it is disabled
enable serial programming	Allows Quickstep programs to be downloaded to the controller via the serial port (default).
enable sntp monitor	Enables UDP debugger printouts of SNTP packet information. Typically used to verify operation by technical support. By default it is disabled.

Command	Description

get COMM#	Returns the current settings of the specified communications port:
	`BlueFusion/>get COMM1`
	`19200, 8, None, 1, CTC Binary, 2`
	`BlueFusion/>`
	Specified as baud rate, data bits, parity, stop bits, protocol, and protocol address (if used).

get Ethernet mode	Return current Ethernet port mode settings. NVSetting is what the non-volatile setting current is:
	`BlueFusion/>get Ethernet mode`
	`Current: 100/FULL NVSetting: AUTO`
	`BlueFusion/>`
	Above shows a connection at 100M full duplex, with auto-negotiate as the requested power-up setting.

get groupname	Returns the current auto-discovery group name assigned to the controller. None by default.
	`BlueFusion/>get groupname`
	`Group Name = NONE`
	`BlueFusion/>_`

get Network	Reserved for WebMON. Returns the current network settings:
	`BlueFusion/>get network`
	`CTC_BF_10063261, 12.40.53.133, 255.255.255.0, 12.40.53.204, 2, 0, true`
	`BlueFusion/>_`
	DNS Name, IP address, subnet mask, gateway, Modbus Address, CTC Node, dhcp enabled if true.

get POP3	Reserved for WebMON. Returns current Email server settings, default is disabled:
	`BlueFusion/>get pop3`
	`0.0.0.0, 110, 10000, 2000, NONE, NONE, false`
	`BlueFusion/>`
	Email server, port, poll rate in milliseconds for checking mail, host time out in milliseconds, user name, password, enabled if true.

get security	Reserved for WebMON. This command returns the current security settings for restricted IP access that is stored in EEPROM. Factory defaults of no restriction are shown below:
	`BlueFusion/>get security`
	`0.0.0.0 0.0.0.0 0`
	`0.0.0.0 0.0.0.0 0`
	`0.0.0.0 0.0.0.0 0`
	`0.0.0.0 0.0.0.0 0`
	`*`
	`BlueFusion/>_`

get SNTP	Reserved for WebMON. Returns current SNTP server settings for time synchronization over the network. The default is enabled and is defined as below:

Command	Description

	`BlueFusion/>get sntp`
	`192.43.244.18, 123, 86400, 0, true`
	
	`BlueFusion/>`
	IP address, port, updated rate in seconds, offset in seconds from GMT, enabled if true.

get systemname	Displays the current system name to be registered with DHCP.
	`BlueFusion/>get systemname`
	`System Name = CTC_BF_10063261`
	
	`BlueFusion/>`
	If name has not already been define the serial number will be preceded by 'CTC_BF_' and registered with the host, as it is shown in this example. Use the 'set systemname' command to set your own name.

get throttles	Displays detailed information with regards to connection throttling settings. Connection throttling allows you to balance the load on the controller between response time to a request and allowing other things to run. Typically you want to respond quickly to a request but if another request is immediate you may want to delay in responding. This is important since some network computers will poll as fast as possible, requesting information immediately after a response. In general, this results in performance degradation. Throttling gives the best of both worlds, enabling the controller to respond immediately to a certain number of requests but then delay if the host requests information too rapidly. The get throttles command returns two sets of data in a format that can be redirected to a script file if desired. The first block is the default values for each supported protocol type, and the second block is all currently active connections, if any. The default ip setting = 0.0.0.0, which means all ip addresses.

`BlueFusion/>get throttles`
`# Default throttle values:`
`set throttle conntype=NETWORK protocol=ANY ip=0.0.0.0 port=-1 packetcnt=1 pause=50 inactivity=100`
`set throttle conntype=NETWORK protocol=TCPBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause=50 inactivity=100`
`set throttle conntype=NETWORK protocol=UDPBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause=50 inactivity=100`
`set throttle conntype=NETWORK protocol=CTNETBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause=50 inactivity=100`
`# Active throttle connections:`
 	
`BlueFusion/>_`	

set COMM#	Reserved for WebMON. Sets the specified serial ports parameters.
set groupname <Name>	Reserved by WebMON. The group name is used to categorize 5100 and 5200 controllers for display into a tree type relationship when using the WebMON auto-discovery feature.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td></td><td>
<pre>BlueFusion/>set groupname Manufacturing.BallWinder1
SUCCESS: Group name changed. Requires write of 1 to

BlueFusion/>get groupname
Group Name = Manufacturing.BallWinder1

BlueFusion/>_</pre>
A 1 must be written to register 20096 for the new group name setting to be saved to non-volatile storage.
</td></tr>
<tr><td>set Ethernet mode <Mode></td><td>The Ethernet mode may be set to:
0 - AutoNegotiate (Default)
10 HALF - 10 M Half Duplex
10 FULL - 10 M Full Duplex
100 HALF - 100 M Half Duplex
100 FULL - 100 M Full Duplex</td></tr>
<tr><td>set Network</td><td>Reserved for WebMON. Reference ‘get Network’.</td></tr>
<tr><td>set POP3</td><td>Reserved for WebMON. Reference ‘get POP3’.</td></tr>
<tr><td>set Security</td><td>Reserved for WebMON.</td></tr>
<tr><td>set SNTP</td><td>Reserved for WebMON. Reference ‘get SNTP’.</td></tr>
<tr><td>set systemname <Name></td><td>Sets current system name to ‘Name’. This is the name to be registered with DNS via DHCP, if enabled (IP address = 0). A write of a 1 to register 20096 is required to make this systemname nonvolatile.</td></tr>
<tr><td colspan="2">set throttle {connType=} {protocol=} {ip=} {port=} {packetCnt=} {pause=} {inactivity=}</td></tr>
<tr><td></td><td>Sets the connection throttling parameters for a particular network connection or change the default values used for a particular protocol. See get throttles for a higher level description of throttling.

Parameter settings are as follows; if one is not included on the command line the default will be used:
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>{connType=}</td><td>NETWORK, DEFAULT (ONLY TYPE CURRENTLY SUPPORTED).</td></tr>
<tr><td>{protocol=}</td><td>UDPBINARY, TCPBINARY, CTNETBINARY, or ANY (default)</td></tr>
<tr><td>{ip=}</td><td>ip address of node. Example: 12.40.53.129. If 0.0.0.0 (default) is used then sets default values used for new connections.</td></tr>
<tr><td>{port=}</td><td>unsigned short value 1024 to 65536, -1 means any port (default)</td></tr>
<tr><td>{packetCnt=}</td><td>number of packets/requests that can be requested consecutively; if -1 (default) then use default value for protocol type selected.</td></tr>
</table>
</td></tr>
</table>

Command	Description
	{pause=} milliseconds to pause after a burst; if –1 (default) then use default value for protocol type selected.
	{inactivity=} idle time before reset consecutive packet/request counter; if -1 (default) then use default value for protocol type selected.

Below is an example of changing the default "pause" time for TCPBINARY protocol connections from the default of 50ms to 175ms. Note that the protocol type must be included when changing the defaults. Protocol ANY is only for setting an actual ip address, not the default:

```
BlueFusion/>get throttles
# Default throttle values:
set throttle conntype=NETWORK protocol=ANY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
set throttle conntype=NETWORK protocol=TCPBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
set throttle conntype=NETWORK protocol=UDPBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
set throttle conntype=NETWORK protocol=CTNETBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
# Active throttle connections:

BlueFusion/>set throttle ip=0.0.0.0 protocol=TCPBINARY pause=175

BlueFusion/>get throttles
# Default throttle values:
set throttle conntype=NETWORK protocol=ANY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
set throttle conntype=NETWORK protocol=TCPBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause= 175 inactivity=100
set throttle conntype=NETWORK protocol=UDPBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
set throttle conntype=NETWORK protocol=CTNETBINARY ip=0.0.0.0 port=-1 packetcnt=1 pause= 50 inactivity=100
# Active throttle connections:

BlueFusion/>
```

reset ethernet	The Ethernet connection is reset at the hardware level.

File Commands

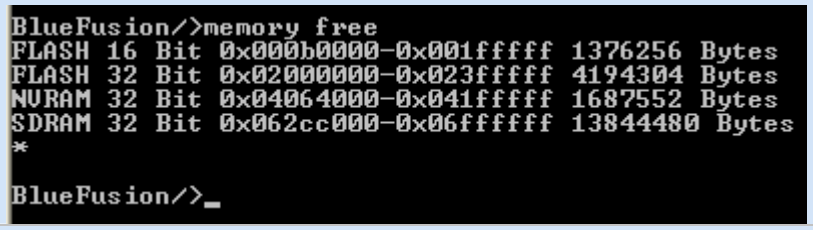
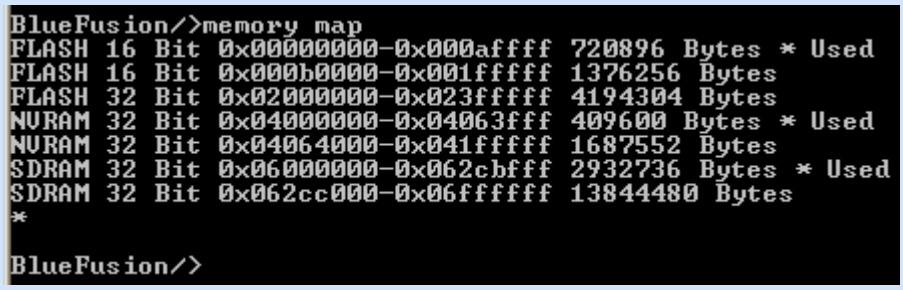
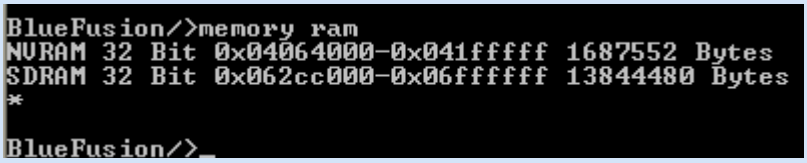
cat <filename>	Lists the specified ASCII file to the display, line by line.

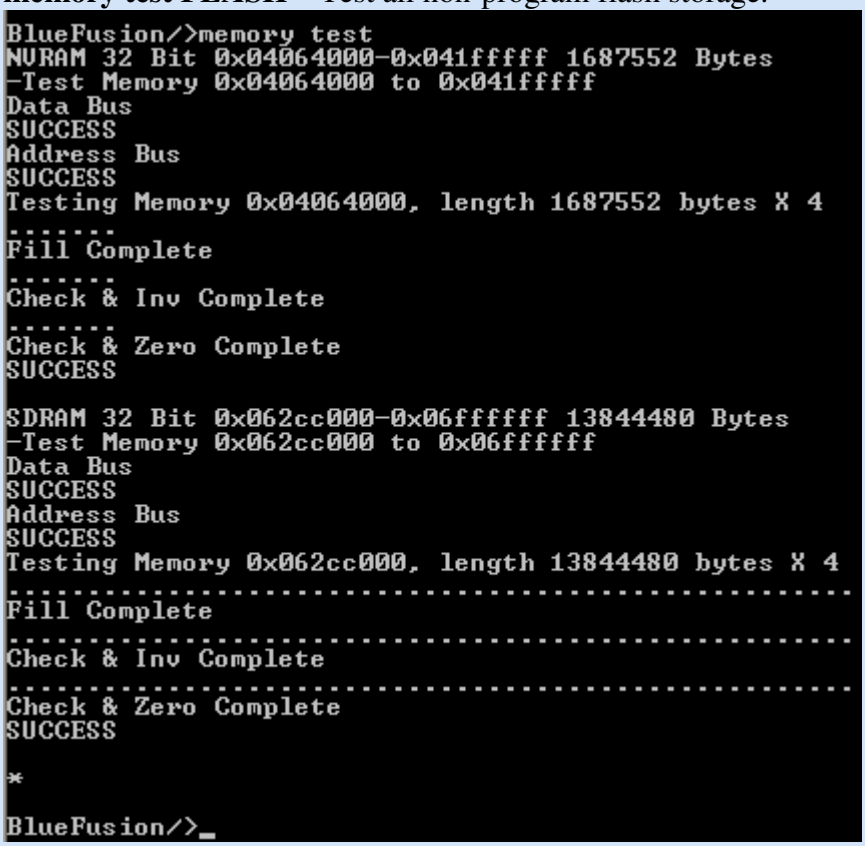
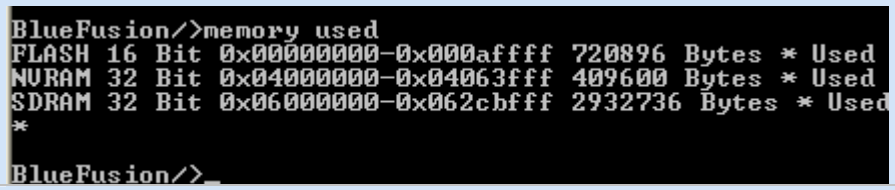
cd	Changes the current working directory:
	`BlueFusion/>cd _system`
	`cd command successful.`
	`BlueFusion/_system/>dir`

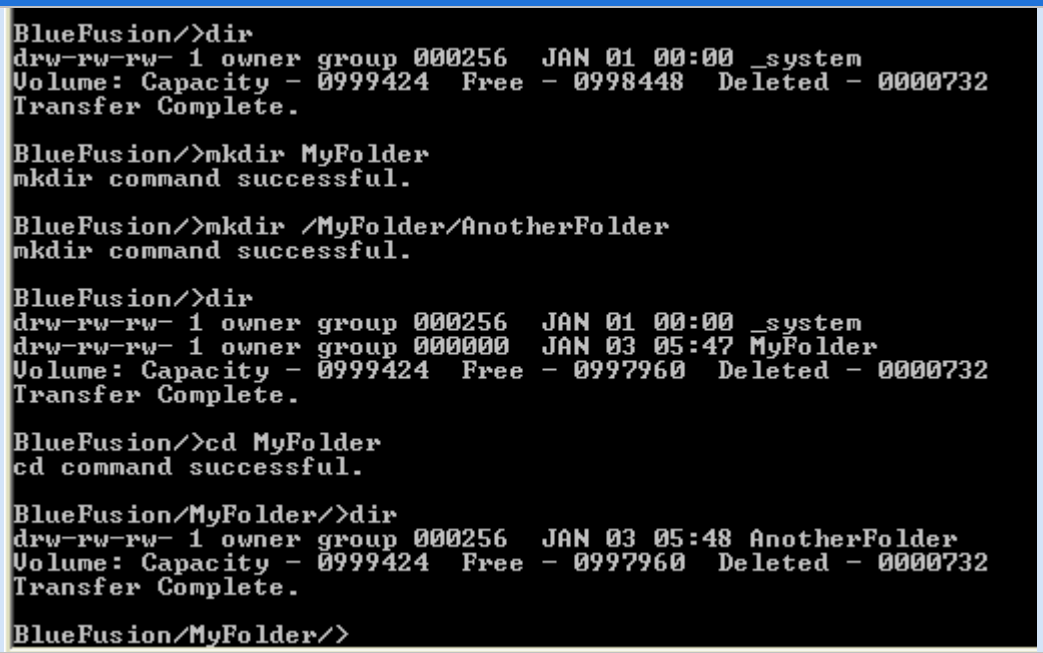
delete	Deletes a file in the current directory. The syntax is: delete filename.

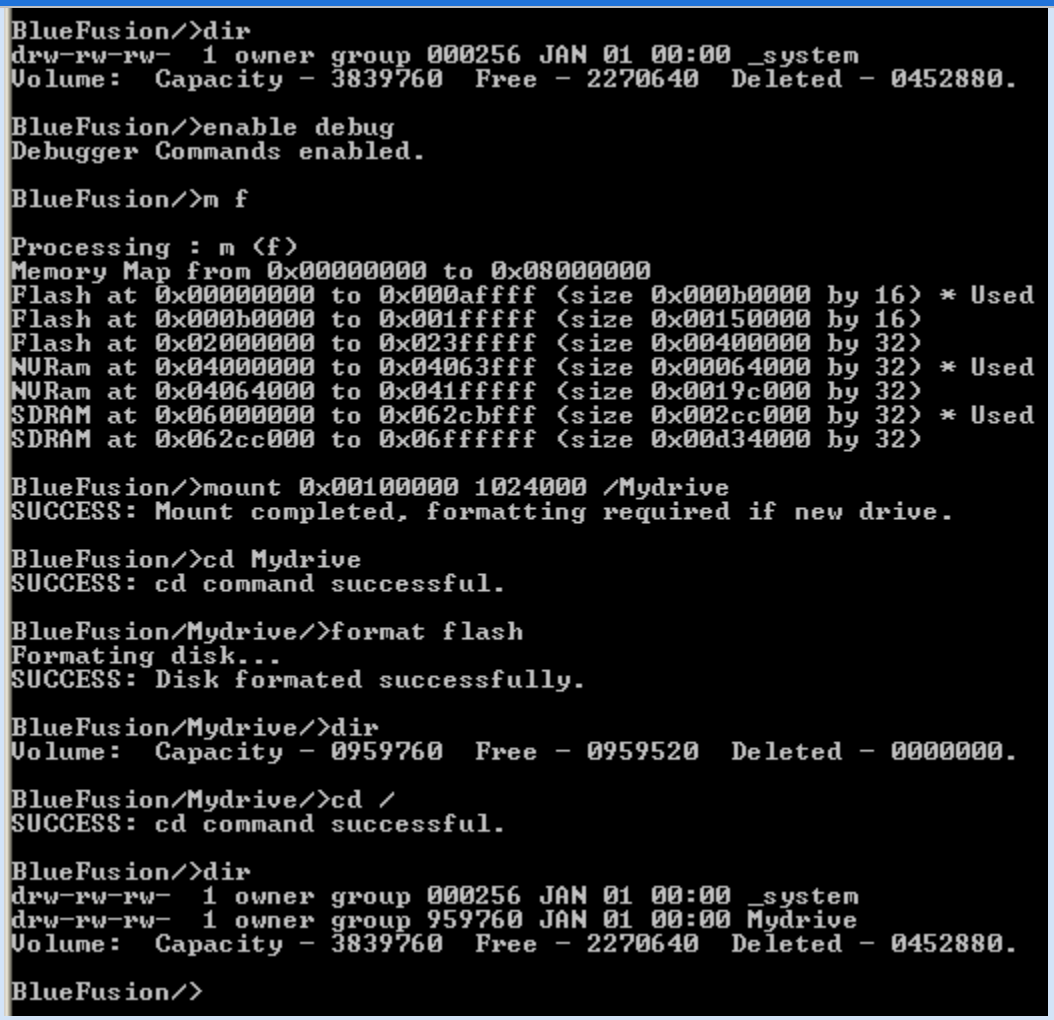
dir	Displays the current directory contents:
	`BlueFusion/_system/>dir`
	`drw-rw-rw- 1 owner group 000256 JAN 01 00:00 Datatables`
	`drw-rw-rw- 1 owner group 000000 JAN 01 00:00 Firmware`
	`drw-rw-rw- 1 owner group 000000 JAN 01 00:00 Messages`
	`drw-rw-rw- 1 owner group 000000 JAN 01 00:00 Programs`
	`drw-rw-rw- 1 owner group 000000 JAN 01 00:00 Scripts`
	`Volume: Capacity - 0999424 Free - 0998448 Deleted - 0000732`
	`Transfer Complete.`
	`BlueFusion/_system/>`

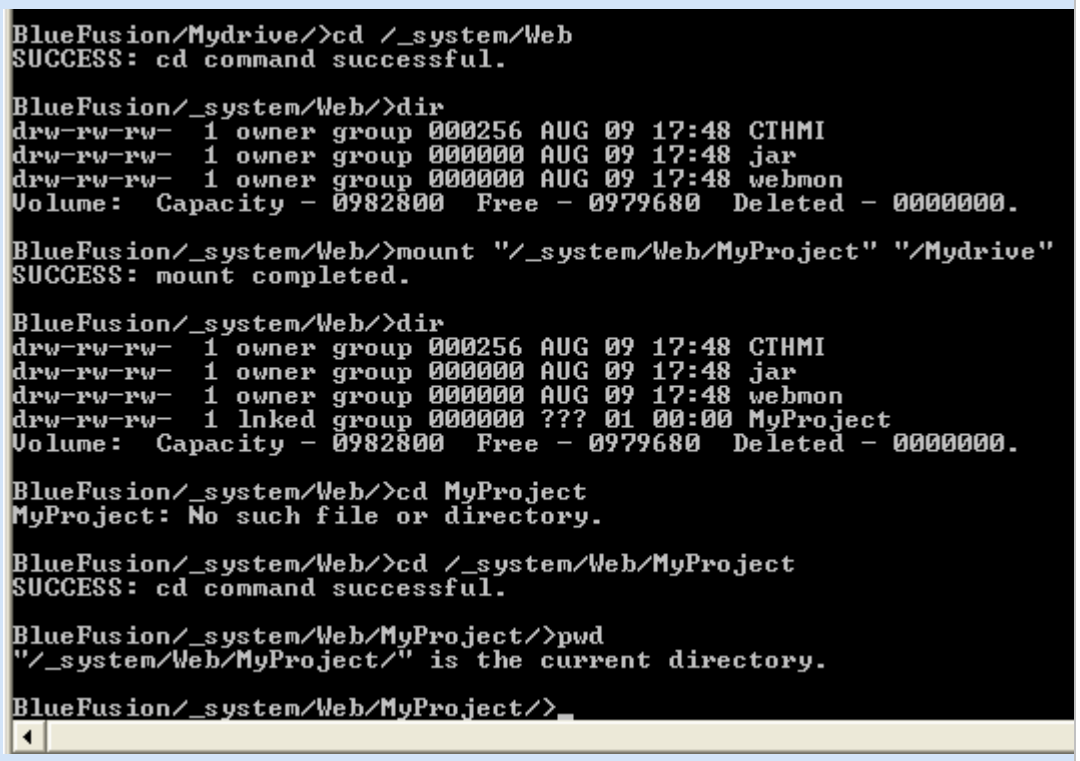
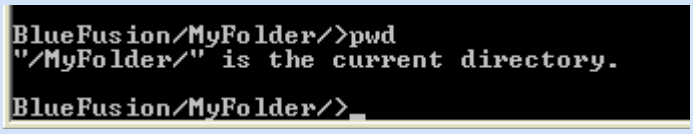
format flash *or* **format disk**	The format flash command is used to fully erase the current flash drive and prepare it for operation.
memory free	Similar to the 'memory map' command, this command only displays memory that

Command	Description
	is available for User disk storage. 
memory map	A complete memory map of all installed memory within the controller is shown, along with what is being used by the 5200 controller itself. Each line which ends with a "* Used" represents an area currently used by the Quickstep firmware. Lines that do not contain "* Used" are available for User disk storage. Reserved area's for future expansion are not marked but appear within the 'System Memory Map' section of the 5200 *Remote Administration Guide, Document 951-520001*. Both FLASH and RAM may be used for disk storage (RAM is battery backed). SDRAM will not survive power down and although available it is not recommended for anything but temporary storage. It is also the area used by 'C' loadable programs and FTP file transfers. 
memory ram	Similar to the 'memory map' command, this command only displays NVRAM and SDRAM memory that is available for User disk storage. 
memory test	This command conducts extensive diagnostics on the memory region returned by the 'memory free' command. This is destructive testing and no RAM disk should be mounted during this test or 'C' program running. The flash drive **must** be re-formated after this test and power cycled. Also the controller is set to 'factory defaults' after execution. Individual memory tests of specific types are also available: **memory test RTC** – Test Real Time Clock internal memory

Command	Description
	memory test E2PROM – Test serial e2 memory and reset to factory defaults (DHCP, default flash disk, etc…). **memory test RAM** – Test only RAM and SDRAM. **memory test FLASH** – Test all non-program flash storage.  Should an error occur the line will begin with "ERROR: <failure message>".
memory used	Similar to the 'memory map' command, this command only displays memory that is used, **not** available for User disk storage. 
mkdir	Creates a new directory/folder in the current directory. A path other than the current one may also be given.

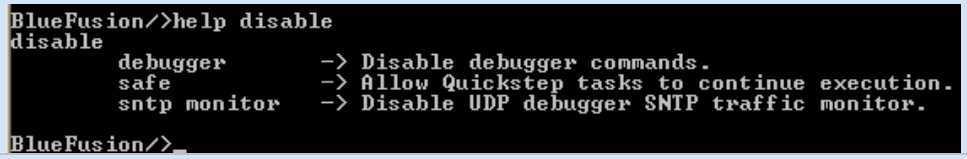
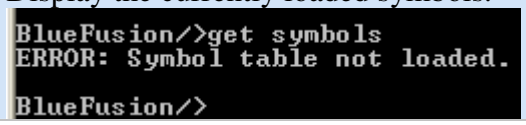
Command	Description
	
mount [Address] [Size] [Name]	**Address** – This is the Hex address the disk is to be created within memory. It must start with '0x' and consist of 8 hex characters following. For example: 0x02000000, specifies the start of the optional 32 bit wide flash disk on the main system board. **Size** – Size is the maximum number of bytes, on a 32K boundary, that the disk should occupy. If a 32K boundary is not specified the firmware will ensure the boundary is met before creating. Note that the larger the size the longer the mount command may take to execute since it is sometimes automatically done. About 45 seconds for a 4M drive. **Name** – The name of the new drive, preceded by a '/'. To create a new drive called Mydrive the name would be '/Mydrive'. The '/' designates it is mounted off of root, (required). Example: Create a drive using the 4M optional main memory flash disk components, starting at memory address 0x00100000. This is to be called Mydrive:

Command	Description
	
mount ["Virtual Path"] ["Actual Path"]	**"Virtual Path"** – The Virtual Path is the path that will be referenced by the accessing host or user, its last node in the directory tree given should not exist. The higher level directories must exist. In other words if the virtual path is "/_system/Web/remapped" then _system and Web must exist but remapped should not as it will be mapped to an actual location by referencing the "Actual Path". All access must reference root ('/') for the virtual directory to function correctly and the double quotes are required. **"Actual Path"** – The Actual Path is the path that will be substituted for all Virtual Path references. This path may reside on the same or a different drive regardless of media type (RAM, SDRAM, or FLASH), or media width (16 or 32 bits). If a mounted drive is being accessed (other than '/') at least a single subdirectory must exist on that drive. The subdirectory does not have to be used nor referenced, just exist. Example: Create a virtual subdirectory under the /_system/Web subdirectory that reference the new drive /Mydrive created in the previous section. The virtual

Command	Description
	directory will be called MyProject and it will use the entire /Mydrive storage area.  In the example, the /_system/Web directory was made the current directory and a list of existing sub-directories was made. Note that MyProject does not exist. The mount command was then executed and another directory run, MyProject now appears. Instead of 'owner', 'lnked' now appears and ??? is listed as the date. Attempting to change to that directory initially fails. As noted previously, all references must be from the root directory, thus including the '/' reference. The command is now done again, referencing the entire path '/_system/Web/MyProject' and that directory now becomes current. The 'virtualdirmap' command can be used to retrieve a list of directory mappings.
pwd	Displays the current path. 
rmdir	Removes a directory contained within the current directory.
umount <driveinfo>	Un-mount a mounted drive, removing it from use. Use the 'mount' command with no parameters to determine what is mounted. A 1 must be written to register 20096 to update non-volatile storage with the change.
umount <"virtual dir">	Un-mount a directory that has been remapped (dirmap) and removed it from use.

Command	Description
<”actual dir”>	The double quotes around each of the directory references are required. A 1 must be written to register 20096 to update non-volatile storage with the change.
virtualdir map	The ‘virtualdir map’ command is used to list all active virtual directory mappings created with the mount command.
Miscellaneous Commands	
#	Comment – Lines beginning with a # sign are ignored and treated as comment lines.
disable debug	Turns the low level debug functionality off. Typically this is reserved for CTC personnel.
disable safe	Allows Quickstep programs to run again after they have been placed in a "safe state" with the enable safe command. Programs are resumed with the first task being allowed to run from where it left off.
enable debug	Turns the low level debug functionality on. The low level debugger allows you to monitor the state of various program threads, view logs and thread logs, read/write bytes of memory, check the state of program mutexes, etc. These are typically things only CTC personnel would be interested in while trying to diagnose a problem remotely, so an detailed explanation of the commands is not given. Reference the ‘C’ User’s Programming Manual for further information. Command summary:
enable safe	Sets a program to a safe stopped state. A flag is set that allows Quickstep to finish

```
BlueFusion/>help debug

<** Diagnostic Functions **>

list -> (l) Display all events
Logmask (mask) (Set Event Log mask)
  Display and set the log mask value
    App(1000), Hw(1), Io(8), Tim(4)
    OS(2), Servo(20), Reg(200), Prof(100)
    Network(10), Protocol(40), Link(400)
L X  -> Clear event log
os (Operating System)
  a  -> all tasks
  c  -> counters
  p  -> pools
  m  -> mutexes
  q  -> queues
  s  -> semaphores
  tn -> task n
  u  -> memory Used

rtc (Real Time Clock)
  R (ad)        -> Read at address
  W ((ad),dat) -> Write at address
  B ((ad),len) -> Block Read at address
  T             -> Time (up to 1 minute)
  Z             -> Reset Chip

memory (read/write memory)
  Enter address and length
  Add a D or d for decimal display
  Add a W or w for word access display

BlueFusion/>_
```

Command	Description
	executing its present instruction, at which point communications is allowed to continue but the program is not. Typically it is best to do a set restart to restart the programs after being placed in a safe state. Optionally disable safe may be executed to resume where you left off (begins with the first task running again from where it left off). safe state is required for loading new data tables, etc. At present, most commands automatically invoke the safe state prior to execution so it is not necessary to use this command. Reference those commands directly to see if a safe state is not automatically invoked.
help	A complete help screen is available via the help command. Individual help for a different group may be invoked by typing `help` followed by the group name. For example, `help set` would display: 
load symbols <filename>	Load the specifed symbol file and activate for debugging.
get symbols	Display the currently loaded symbols. 

CHAPTER

3

Script Execution

In addition to being used in FTP and Telnet sessions, script files can be created that execute a sequence of the commands listed in Chapter 2. Script files, when properly named and stored on the Flash Disk, also allow execution to be initiated from within a Quickstep program.

In order to run this way, script names must use the format *Script###.ini* where ### is a number between 001 and 999. Scripts must also reside in the Programs directory on the 5200's flash disk. The script can be executed by writing the script number to Register 12311. To execute the file *Script004.ini* from Quickstep:

```
Store 4 to Reg_12311
```

Script Registers

Script Register 12311: Writing a numeric to this register will cause the corresponding script to be executed. For example, writing a 4 to this register will cause *Script004.ini* to be executed and a 4 to appear when this register is read.

Script Result Register 12312: When this register's value = 1, the script successfully executed, 0, will appear during execution, other values in this register represent error codes.

Example FTP Sessions

ls "!get versions"
< version information displayed>
ls "!set safe"
< Quickstep in Safe Mode>
ls "!update modules"
< results displayed >
ls "!set restart"
< Quickstep restarted>

Example Script file

```
# This is a comment line
# set register 1000 to a 2
set register 1000 = 2          # this is another comment
1000 = 2                       # this is another way to do the same thing
#
# read a bunch of registers, if the script is run with the > Script###.ini command then
# contents of register will be sent to the file for later replay, otherwise the read will
# be done but data thrown away.
# read the contents of 13002 to 13010
get register 13002-13010
# read registers 100 to 500 as a block but using the shorter command
100-500
# run another script file from within this file
run Script002.ini
```

CHAPTER

4

Script Groups

Script Groups define categories of commands that may appear within a script file, including a *_startup.ini* boot file. They may be used to define numerous items relevant to the administration of the terminal, or simply the modification of registers under program control. Three separate Groups are available:

- Network
- Security
- Commands (default if no other group specified)

The type of the Group that is activated (encountered within a script file) will affect the processing of each command within that section. To activate the Group, and hence the commands available within that Group, simply declare it in the script file:

[Network]
……
[Security]
…..
[Commands]
……

A sample file is available in Section 3.4.

Group [NETWORK]

The Network Group is used to define communication address parameters, such as IP addresses, ports, protocols to enable, etc. It is used as a logical text separator within a Script file.

Parameter	*description*
IP_ADDRESS	IP address identifying this 5200 board on the network. Enter each of 4 octets, separated by a dot. A change will only be recognized during

Parameter	description
	power up.
SUBNET_MASK	IP SUBNET MASK OF THE NETWORK THAT WE ARE OPERATING ON. ENTER EACH OF 4 OCTETS, SEPARATED BY A DOT. A CHANGE WILL ONLY BE RECOGNIZED DURING POWER UP
GATEWAY_ADDRESS	IP address that packets are to be sent to if they do not reside on your network. Typically a router address. If none is present enter 0.0.0.0, which is the factory default. A change will only be recognized during power up.
CTCNET_DEVICENODE	Single byte, unique address that this 5200 is to be known by on the network if the CTNET protocol is to be used. By factory default it is set to 0, not enabled. A change may be made dynamically. A zero, "0", will disable the protocol and conserve CPU resources.
BINARYUDP_SERVER_PORT	IP port that the 5200 should listen for CTC Binary Protocol UDP request packets on. By default it is set by the factory to port 3000. A change will only be recognized during power up. A zero, "0", will disable the protocol and conserve CPU resources.
BINARYTCP_SERVER_PORT	IP port that the 5200 should listen for CTC Binary Protocol TCP request packets on. By default it is set by the factory to port 6000. A change will only be recognized during power up. A zero, "0", will disable the protocol and conserve CPU resources.
PEERUDP_SERVER_PORT	IP port that the 5200 should listen for CTC Peer-to-Peer register request packets on. By default it is set by the factory to port 4000. A change will only be recognized during power up. A zero, "0", will disable the protocol and conserve CPU resources.
MODBUSTCP_SERVER_PORT	IP port that the 5200 should listen for Modbus TCP register command packets. By default it is set by the factory to port 502. A change will only be recognized during power up. A zero, "0", will disable the protocol and conserve CPU resources.

Group [SECURITY]

The SECURITY Group restricts access to the 5200 controller. You can toggle certain communication protocols ON/OFF and can enter ranges of IP addresses for the different services that are available. As a security measure, these IP addresses are checked before access is granted to a resource. The standard method of requesting a username and password is not required. Multiple entries are allowed and each is checked until permission is granted or no more entries exist. Note that if a [SECURITY] Group header is included in the file, all network access is disabled, unless specifically enabled. Not including the header means all access is enabled.

An entry consists of an IP address (or range of IP addresses) followed by a list of what is allowed (a blank entry indicates that everything is allowed). The default setting allows all accesses when security is not defined.

Each IP Address has the following options that you can enable or disable:

PARAMETER	description
TELNET	ALLOWS TELNET ADMINISTRATION ACCESS.
PEERUDP *or* **PEER-UDP**	Allows peer-to-peer communications using UDP.
BINARYTCP *or* **CTC-TCP**	Allows binary protocol TCP communications.
BINARYUDP *or* **CTC-UDP**	Allows binary protocol UDP communications.
CTCNET	Allows certain nodes to have access using the CTNet protocol. It is defined by setting the first three octets to 0 and the last octet to the allowable address (0.0.0.#). Place at the beginning of the list to speed up access. The default setting is that all nodes are allowed.
FTP	ALLOWS FTP FILE ACCESS.
MODBUSTCP *or* **MODBUS-TCP**	Allows Modbus TCP client connections.
RAWSOCKET *or* **RAW-TCP**	Allows raw socket connections via TCP.
HTTP *or* **WEB**	Allows access to the 5200 Web Server

The following examples show how these options work:

Examples:

```
0.0.0.10  0.0.0.20        #  Allow CTNET nodes 10 to 20 to have access.
208.164.187.27            #This allows all access, no restrictions from this IP Address
208.164.187.25  FTP  #FTP access from this IP Address
208.164.187.240 208.164.187.250  PEERUDP  #Only peer comm. allowed in this
range
12.40.53.001 12.40.53.255         # Range of IP addresses allowed full access
```

Group [COMMANDS]

The COMMAND Group allows you to run the commands defined within the script language inside a file. By default if no section name is given this is the section assumed. Each command is sequentially executed so take care that it is the proper sequence.

For example, to set register 20000 to a CTNET node of 50 enter: 20000 = 50. This does the same thing as defined in the NETWORK group. See the sample `5200.ini` file for further examples in section 3.4.

Sample _startup.ini File

Sample _startup.ini text file, each line terminated with a CR LF:

Note: [NETWORK] and [SECURITY] sections are not needed unless you wish to override the default settings. For example, it is preferred to set the IP address using DHCP and not within the .ini file. If you include the [SECURITY] sub-section then all network access to the controller is disabled (default is fully enabled when section not included), unless specifically enabled by an included [SECURITY] sub-section command.

```
[NETWORK]
#No Network settings

[SECURITY]

# If below is left undefined then all IP addresses are allowed with all permissions
#Otherwise each Address can be limited to access permissions
# IP address may be followed by any of the following, if blank all access is allowed
# else if anything is specified only that listed is allowed
#       FTP - Allows FTP access
#       TELNET – Allows Telnet access
#       PEERUDP - Allows peer to peer communications via UDP
#       BINARYUDP - Allows binary protocol UDP communications
#       BINARYTCP - Allows binary protocol TCP communications
#       MODBUSTCP - Allows Modbus access protocol TCP Communications
#208.164.187.27          #This allows all access (supervisor, no restrictions)
#208.164.187.25  FTP
#208.164.187.250 208.164.187.240  PEERUDP  #Only peer comm. is allowed
```

```
#208.164.187.27                 # Unrestricted access available to this address
12.40.53.001 12.40.53.255       # Range of IP addresses allowed general access
```

[COMMANDS]

```
# Registers are initialized in the order given, values are in decimal
# Any block not monitored will be filled in with zero's.  Add other Peer blocks as
# required
# Controller 1
21005 = 4          # Set the number of registers in block first for allocation
21000 =  12           # Now set peer IP address
21001 = 40
21002 = 53
21003 =27
21008 = 1003
21009 = 2       # Set protocol to Modbus Master
21008 = 0       # Set back to way was
21004 = 8001          # Set the register to start monitoring
21006 = 50            # Set number of milliseconds between updates
# Controller 2
21015 = 4          # Set the number of registers in block first for allocation
21010 =  12           # Now set peer IP address
21011 = 40
21012 = 53
21013 =245
21018 = 1003
21019 = 2       # Set protocol to Modbus Master
21018 = 0       # Set back to way was
21014 = 8001          # Set the register to start monitoring
21016 = 50            # Set number of milliseconds between updates
# It is even possible to monitor our own registers and have them available for both local
# and remote reference.  Below a number of IO points are monitored.  The register
# list is also that required for the special Binary Protocol TCP System IO command 0x53.
#Analog Out
21025 = 128          # Set the number of registers in block first for allocation
21020 =  12          # Now set peer IP address to ourself
21021 = 40
21022 = 53
21023 = 219
21024 = 8001     # Set the register to start monitoring
21026 = 50       # Set number of milliseconds between updates, writing timer starts poll
#Analog In
21035 = 128      # Set the number of registers in block first for allocation
21030 =  12      # Now set peer IP address to ourself
21031 = 40
21032 = 53
```

```
21033 = 219
21034 = 8501       # Set the register to start monitoring
21036 = 50         # Set number of milliseconds between updates, writing timer starts poll
#Digital Out
21045 = 10         # Set the number of registers in block first for allocation
21040 =  12        # Now set peer IP address to ourself
21041 = 40
21042 = 53
21043 = 219
21044 = 10001      # Set the register to start monitoring
21046 = 50         # Set number of milliseconds between updates, writing timer starts poll
#Digital In
21055 = 10         # Set the number of registers in block first for allocation
21050 =  12        # Now set peer IP address to ourself
21051 = 40
21052 = 53
21053 = 219
21054 = 11001      # Set the register to start monitoring
21056 = 50         # Set number of milliseconds between updates, writing timer starts poll
#Regs 901 to 1000
21065 = 100        # Set the number of registers in block first for allocation
21060 =  12        # Now set peer IP address to ourself
21061 = 40
21062 = 53
21063 = 219
21064 = 901        # Set the register to start monitoring
21066 = 50         # Set number of milliseconds between updates, writing timer starts poll
```

CHAPTER

5

Formatted Messaging

The 5200 can transmit string-formatted messages, similar to the format supported by the 'C' function 'sprintf'. Each message may consist of just text and/or embedded references to any number of registers, whose values will be substituted just prior to transmission. Message format definitions are stored as records in a file called *message.ini* which is located in the */_system/Messages* subdirectory of the flash disk. Each line of *message.ini* is considered a record, from 1 to a maximum of 50 messages.

Messages are written to the default communications port set in register 12000, which is the standard Serial port selection register in Quickstep. Writing to the ***Message String Transfer Register*** (12316) selects which record to dynamically format and write out the communications port. A read returns the status of the write, with 0 meaning success. The 5200 supports up to 7 communication ports, two of which are dedicated to RS232, while the remaining 5 are assigned by the program as bidirectional TCP redirector ports. The redirector ports appear to Quickstep as RS232 ports, but actually either connect to a remote terminal server or host based application program

Typically a message consists of text with a 'sprintf" formatted specification, followed by `r####`, where `####` is the desired register. Therefore, to read register 8501 to be exactly 5 characters with preceding 0's, `%05dr8501` would be inserted in the text string. Note the `%05d` is the same as a 'printf'/'sprintf' and actually uses the exact same function, only enhanced. This means `%05Xr8501` would cause hexadecimal values to be generated. Sample strings using the previous example could be entered in the *message.ini* file as:

Analog Value = %05dr8500\r\n
Analog Value = %05dr8501\r\n

If the above are the only two entries in the *message.ini* file, then writing a 2 to the ***Message String Transfer Register*** would cause the second line to be processed and the following to be written out the RS232 port if a 583 were in register 8501:

Analog Value = 00583<CR><LF>