Model 5100
Script Language
Guide

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
5101/5102/5103/5104	B, C, and E	>=4.05.54

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.

TABLE OF CONTENTS

1.0 Overview

As an enhancement to Quickstep, a Script Language has been added that allows the performance of many administrative tasks from within the language. The 5100 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. Some of the functionality available is:

Under Quickstep Program Control:

1. Save and load data tables from a resident flash disk (not implemented in V4.05).
2. Save sets of registers and restore from flash disk. Saving dynamically creates a new script file for later playback.
3. Load alternate Quickstep programs for execution.
4. Ability to run script command files (.ini files). Nested scripts are also allowed.

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. Advanced remote diagnostics.

1.1 Script File Naming/Access

Script files are text files consisting of various commands as defined in the "Script Command" section of this document. 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 reference 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 a 1 if successful, else an error code.

1.2 Boot Script

At power up and system reset a script file by the name of 5100.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 sections. A reset will be required to activate any

IP address informational changes, ie., ip, subnet mask, or gateway, refer to Section 3.0. Beware that if you change any IP information using CTCMON and the 5100.ini file also modifies that information, at the next power cycle the 5100.ini file will override it. Should you wish to remove the 5100.ini file, other than 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.

1.3 Data Table File Naming/Access (not implemented in V4.05)

Data table files are in a special binary format, allowing for fast transfer and are not compatible with Quickstep Program generated files. All data table files must reside within the flash disk directory /_system/Datatables and are of the naming convention Datatable001.dat, Datatable002.dat, … The 001, 002… numeric part of the file name reference the value that must be written to the ***Data Table Transfer Register*** (12313) in order for that table to either become the current table or the values of the current table to be written to it. Writing a 1 to the ***Data Table Transfer Register*** causes the current data table to be written to Datatable001.dat. Writing a 1001 to the ***Data Table Transfer Register*** causes the contents of Datatable001.dat to be transferred to memory, the contents becoming the current data table. Upon execution, a read of the ***Data Table Transfer Register*** will return the number of the last data table loaded. A read of the ***Data Table Result Register*** (12314) will return a 1 if successful, else and error code.

1.4 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, via the ***Script Register*** write, the script is run to completion, atomic (completely within) to 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 a '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.

2.0 Script Language

2.1 Commands

Note: The screen examples shown for the commands described below are from a PC telnet session, refer to the section discussing telnet for more detail. They 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". Refer to FTP for more detail. Typically either telnet is used or the commands are stored on a single line within a text .ini file.

2.1.1 disable

disable debug

Turns the low level debug functionality off. Typically this is reserved for CTC personnel.

disable safe

Allow 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.

Figure 2.0: disable safe command

disable sntp monitor

Disables UDP debugger printouts of sntp packet information. By default it is disabled.

2.1.2 enable

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 thread logs, read/write bytes of memory, check the state of program mutexes, etc. Typically things only CTC personnel would be interested in while trying to diagnose a problem remotely, hence a detailed explanation of the commands is not given other than its help screen, which may be of interest to some:

```
<** 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

(* = not implemented)
[quit or logout to exit telnet session]

BlueFusion/>_
```

Figure 2.1: enable debug command

enable safe

Set a program to a safe stopped state. A flag is set that allows Quickstep to finish 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.

```
BlueFusion/>enable safe
Quickstep programs stopped, safe mode enabled.

BlueFusion/>_
```

Figure 2.2: enable safe command

enable sntp monitor

Enables UDP debugger printouts of sntp packet information. Typically used to verify operation by technical support. By default it is disabled.

2.1.3 get

get datatable Datatable###.dat (not implemented)

Loads the named Quickstep data table from the flash disk directory /_system/Datatables, making it the active data table. It is suggested that the 'set safe' command be executed prior to this command to ensure the Quickstep program is not accessing the current data table. Note that the .dat image is a binary file with a special format, designed for fast, direct loading and replacement of the existing data table. It is not the same format output by the Quickstep development system. This command may be executed directly within a script, via the command line of FTP/Telnet, or by writing 1000 + the number ### specified within the file name, to the ***Data Table Transfer Register***. For example to load Datatable002.dat, write a 1002 to the ***Data Table Transfer Register***.

get date or date

Display the settings of the real-time clock, example:

```
BlueFusion/>get date
Current Date/Time = Thursday, 01/03/2000 04:38:47

BlueFusion/>
```

Figure 2.3: get date or date command

get register #####

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. Reference the section below detailing script recording capabilities. To read a block use the syntax #####-#####. You may also list other registers, using a comma delimiter, for example #####-#####, #####-#####, ##### … 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/>_
```

Figure 2.4: get register command

get status

Displays the current program status during an FTP session.

```
BlueFusion/>get status
Program Status = FAULT

BlueFusion/>
```

Figure 2.5: get status command

get systemname

Displays the current system name to be registered with DHCP.

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.

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, 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, the second block are all currently active connections, if any:

```
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:
set throttle conntype=NETWORK protocol=TCPBINARY ip=12.40.53.168 port=4885 packetcnt=1 pause=50 inactivity=100

BlueFusion/>_
```

Figure 2.6: get throttles command

Note that for the default values the "ip=0.0.0.0" setting.

get versions

Displays the current firmware revision levels, serial number, and MAC Address of the unit, along with what modules reside in what slots and supported thermocouples, if any.

```
BlueFusion/>get versions

5100 Serial Number = 00054391  MAC Address = 00C0CB00D477
Base Firmware Revisions:
  Quickstep SH2 Application          V04.04.03
  Quickstep SH2 Monitor              V02.68
  Atmel Serial Comm                  V01.09

Slot Firmware Revisions:
  1. 5110 Digital 8 Input            V00.00
  2. 5120 Digital 8 Output           V00.00
  3. Empty                           V00.00
  4. Empty                           V00.00
  5. Empty                           V00.00
  6. 5140 Dual Servo                 V01.158


BlueFusion/>
```

Figure 2.7: get version command

2.1.4 help

A complete help screen is available via the help command. Individual help for different group may be invoked by typing help followed by the group. For example “help set” would display:

```
BlueFusion/>help set
set
*   datatable <filename>    -> Save datatable to /_system/Datatables/<filename>
    date Tuesday, 12/6/2003 8:55:00 -> Set real time clock time/date (24hr)
*   password <password>     -> Set new password to restrict access
    register <####> = <###> -> Set register values
    Examples: set register 13002=0
              set register 13002-13006=0, 20000=5, 20005-20010=12
    reset                   -> Set Quickstep reset System state.
    restart                 -> Set Quickstep restart System state
    running                 -> Set Quickstep running System state
    stop                    -> Set Quickstep restart System state
    throttles               -> Set comm. throttling to control performance
<#####>=<##>                -> same as 'set register'

BlueFusion/>
```

Figure 2.8: help set command

2.1.5 set

set datatable ‘Datatable###.dat’ (not implemented)

Writes the existing data table, in memory, to the specified data table file on the flash disk within the subdirectory /_system/Datatable. Command is executed directly from within FTP/Telnet, a script file, or by a Quickstep program writing ### to the ***Data Table Transfer Register***.

set password

Sets the current 'admin' user id password for telnet and ftp access. A write to of a 1 to register 20096 is required to make this password nonvolatile.

set systemname 'Name'

Sets current system name to 'Name'. This is the name to be registered with DNS via DHCP, if enabled (IP address = 0). A write to of a 1 to register 20096 is required to make this systemname nonvolatile.

set register ##### = #####

Sets a register to a particular value. A block may be set to a particular value by using #####-##### as a range.

Figure 2.9: set register command

set reset

Sets the current Quickstep program state to RESET.

set restart

Restarts the current Quickstep program.

set running

Resume the execution of Quickstep programs, from a stopped state.

set stopped

Sets a Quickstep program to the stopped state.

set throttle {connType=} {protocol=} {ip=} {port=} {packetCnt=} {pause=} {inactivity=}

Set the connection throttling parameters for a particular network connection or change the default values used for a particular protocol. Reference "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:

PARAMETER	*DESCRIPTION*
{connType=}	NETWORK, default (only type currently supported).
{protocol=}	UDPBINRY, TCPBINARY, CTNETBINARY, or ANY (default)
{ip=}	ip address of node, Example: 12.40.53.129, if 0.0.0.0 (default) then sets default values used for new connections.
{port=}	unsigned short value 1024 to 65536, -1 means any port (default)
{packetCnt=}	number of packets/requests that can be requested consecutively, if -1 (default) then use default value for protocol type selected.
{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/>
```

Figure 2.10: set throttle command

2.1.5 *reset*

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.

2.1.6 run

run script Script###.ini

Execute the named script. The script file must be resident on the flash disk in the /_system/Scripts subdirectory.

run script Script###.ini > Script###.ini

Execute the named script and any registers that are read, their contents 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 alternate get/set command defined below.

run program qsprogram.dso

Loads the named Quickstep program for execution from the flash disk directory /_system/Programs and begin 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.

2.1.7 update

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 *5100 Remote Administration Guide* for details of filename conventions.

Example: update com5100V0109.bin

2.1.8 Register Read and Write

#####

Read the specified register, same as 'get register' command. To read a block of registers use '#####-#####', for example to read 1000 to 1050, 1000-1050 would be entered. You may also list other registers, using a comma delimiter, for example #####-#####, #####-#####, ##### …

=

Write a value to the specified register, same as the 'set register' command. 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, 1000-1050 = 0 would be entered.

2.1.9 File commands

cd

Change the current working directory:

```
BlueFusion/>cd _system
cd command successful.

BlueFusion/_system/>dir
```

Figure 2.11: cd command

delete

Delete a file in the current directory. Syntax – delete filename.

dir

Display 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/>
```

Figure 2.12: dir command

mkdir

Create a new directory/folder in the current directory; a path other than the current may also be given.

```
BlueFusion/>dir
drw-rw-rw- 1 owner group 000256  JAN 01 00:00 _system
Volume: Capacity - 0999424  Free - 0998448  Deleted - 0000732
Transfer Complete.

BlueFusion/>mkdir MyFolder
mkdir command successful.

BlueFusion/>mkdir /MyFolder/AnotherFolder
mkdir command successful.

BlueFusion/>dir
drw-rw-rw- 1 owner group 000256  JAN 01 00:00 _system
drw-rw-rw- 1 owner group 000000  JAN 03 05:47 MyFolder
Volume: Capacity - 0999424  Free - 0997960  Deleted - 0000732
Transfer Complete.

BlueFusion/>cd MyFolder
cd command successful.

BlueFusion/MyFolder/>dir
drw-rw-rw- 1 owner group 000256  JAN 03 05:48 AnotherFolder
Volume: Capacity - 0999424  Free - 0997960  Deleted - 0000732
Transfer Complete.

BlueFusion/MyFolder/>
```

Figure 2.13: mkdir command

pwd

Display the current path.

```
BlueFusion/MyFolder/>pwd
"/MyFolder/" is the current directory.

BlueFusion/MyFolder/>_
```

Figure 2.14: pwd command

rmdir

Remove a directory contained within the current directory.

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

To be run this way, scripts must be named "*Script###.ini*" where ### is a number between 001 and 999. Scripts must also located in the Programs directory on the 5100's flash disk. The script can the be executed by writing the script number to Register 12311. To execute the file `Script004.ini` from Quickstep:

```
Store 4 to Reg_12311
```

2.1.10 printf command

printf

printf <text>. Used to test the output of a formatted string that would be used for the message.ini files. For example:

```
BlueFusion/>printf The contents of the timer register = %d13002.
The contents of the timer register = 3213002.
BlueFusion/>_
```

2.2 File Script Special Registers

2.2.1 Script Register

Script Register: 12311, writing a numeric to this register will cause the corresponding script to be executed, for example: write a 4 and Script004.ini is executed.

Script Result Register: 12312, 1 = successfully executed, else error code (TBD).

2.2.2 Data Table Register (not implemented)

Data Table Transfer Register: 12313, writing a numeric to this register will cause the current data table to be saved to the corresponding script, writing the numeric within the filename + 1000 will cause an existing file within the flash disk to be loaded into memory, becoming the current data table. For example: write a 4 and Datatable004.dat is loaded and its values made the current table. *(to be implemented in a future release)*

Data Table Result Register: 12314, 1 = successfully executed, else error code (TBD).

2.2.3 Flash Disk Space Register

Flash Disk Space Register: 12315, contains the approximate free space available on the flash disk.

2.3 Examples Scripts/Commands

2.3.1 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>

2.3.2 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
# save our existing data table file (could be done by Program without script
# but not with this type of filename, ### not needed in name)
save datatable CurrentRecipe.dat          (not as yet implemented)
#load in a new one
load datatable AlternateRecipe.dat        (not as yet implemented)
```

3.0 5100 Script Groups

Script Groups define categories of commands that may appear within a script file, including a 5100.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)

Based on the Group that is activated (encountered within a script file) will effect 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.

3.1 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 5100 board on the network. Enter each of 4 octets, separated by a dot. A change will only be recognized during 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 5100 is to be known by on the network if the CTCNET 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 5100 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 5100 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 5100 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 5100 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.
MODBUSTCP_CLIENT_PORT	IP port that the 5100 should initiate and Modbus TCP register command packets on. Unlike Server mode, in this case we are the client and expect to be communicating with another host or controller server process. See Section X.X for more information on the use of Modbus Client protocol. 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.
THROTTLE _CONNECTION	These are the parameters with which to throttle a network connection. Some protocols, such as CTNET poll the controller at high speed but many times the data is not needed that quickly or it is only needed in bursts. This parameter allows you to set the number of packets that can be processed within a particular timing window and an inactivity timer to reset the delay for burst mode. THROTTLE_CONNECTION = {protocol} {IP info} {port info} {timing parameters}

3.2 Group [SECURITY]

The SECURITY Group restricts access to the 5100 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	Allows peer-to-peer communications using UDP
BINARYTCP	Allows binary protocol TCP communications.
BINARYUDP	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	Allows Modbus TCP client connections.
RAWSOCKET	Allows raw socket connections via TCP.
TELNET	Allows Telnet administration access.

The following examples show how these options work:

Examples:
0.0.0.10 0.0.0.20 # Allow CTCNET 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

3.3 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 CTCNET node of 50 enter: 20000 = 50, this does the same thing as defined in the NETWORK group. See the sample 5100.ini file for further examples in section 3.4.

3.4 Sample 5100.ini File

Sample 5100.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
```

4.0 5100 Formatted Messaging

4.1 Message String Transmission

The 5100 is capable of transmitting 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. The selection of which record to dynamically format and write out the communications port is by writing to the ***'Message String Transfer Register'*** (12316). A read returns the status of the write, with 0 meaning success. The 5100 supports up to 7 communication ports, two of which are dedicated to RS232 and the remaining 5 assigned by the program as bidirectional TCP redirector ports. The redirector ports appear to Quickstep as RS232 ports, but actually either connects 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 test 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>

Where <CR> = 0x0d and <LF> = 0x0a. Note that writing a 1 would have referenced the first string, thus reading register 8500.

More complicated strings can be created, referencing any register or combination of registers. Note that no completely formatted string may exceed 255 bytes and no record entry in the message.ini file may exceed 132 bytes/line.

4.2 message.ini Extended Formats

As described previously, the 'message.ini' file format is similar in structure to that of the 'C' sprintf function, with additional enhancements. References to registers, data table cells and time/date stamp formats are supported using this extended format:

Register (decimal) - %0#dr<register> or %dr<register>
Example: %05dr13002 (fix size with leading 0's to at least 5 places, reg 13002)

Register (hexidecimal) - %0#xr<register> or %Xr<register>
Example: %05xr13002 (fix size with leading 0's to at least 5 places, reg 13002)

Register (ascii) - %cr<register> or %cr<register>,<length> or %cr<register>,r<register>
Example: %cr12001,r12302 (convert the serial port buffer to ASCII characters)
Example: %cr12001,3 (convert the first 3 serial port buffer registers to ASCII)

Data Table Cell - %0#dD<row>,<col> or %dD<row>,<col>
Example: %05dD1,2 (fix size with leading 0's to at least 5 places, row 1, column 2, from the data table).

Time/Date Stamp - %T!<time/date format>
Example: %T!HH:mm:ss!
%T!MM/DD/YYYY!
Where each below are optional:
HH = hour (24 hour format)
mm = minute
ss = seconds
MM – month
DD – day
YY – year in 2 decimal format, no century.
YYYY – year in 4 decimal format, including century.
E – Day in week, text – Mon, Tue, Wed, Thu, Fri, Sat, or Sun
Z – Time zone information in 5 digit format - <sign>HH:mm from GMT

Note:

- All other characters are treated as filler text, except ending ‘!’.
- Maximum 48 character Time/Date Stamp string.