Register Files

Related Documents:
Register Load and Store Example.qcp
Reg File Adv Example.txt
Reg File Adv Example - Include 2.txt
Reg File Adv Example - Include 3.txt
Reg File Adv Example - Include.csv
Reg File Adv Example.qcp

The document details the use of SilverLode's Register Files System. This is an advanced topic. For general background please see the SilverLode User Manual.

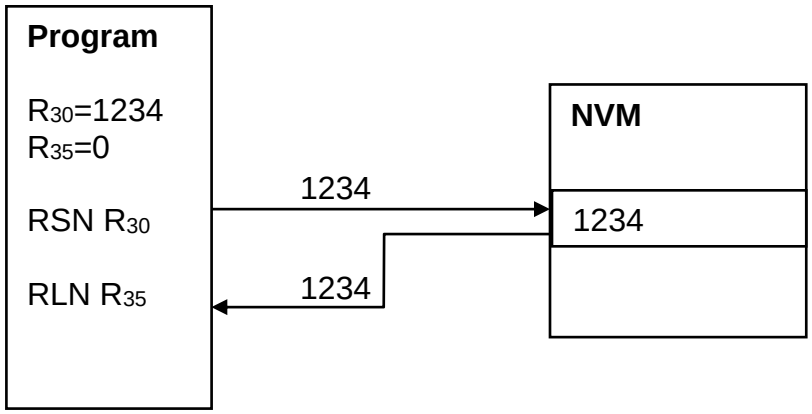
Overview

Register data can be stored to and loaded from Non-Volatile Memory (NVM) using the following command

NVM Commands

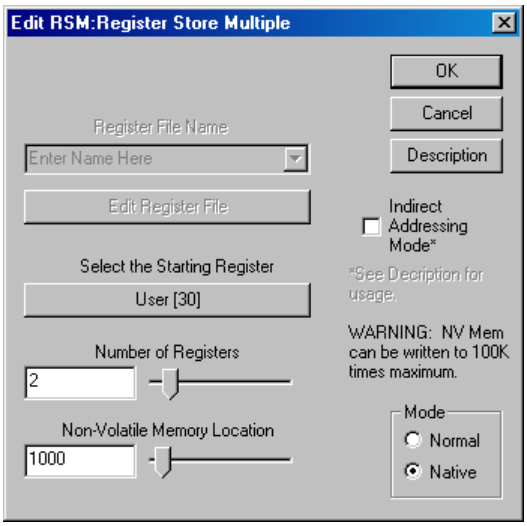
- Register Store Non-Volatile (RSN) – Store data from one register
- Register Store Multiple (RSM) – Store data from multiple registers
- Register Load Non-Volatile (RLN) - Load data from NVM to one register
- Register Load Multiple (RLM) - Load data from NVM to multiple registers

The following example shows RSN being used to store register 30 having a value of “1234” to NVM. RLN is then used to load register 35 from non-volatile memory. At the end of the program, register 35 equals “1234”.



NVM Access

The NVM Commands must specify a Location (address) in NVM where the data is stored. The NVM Location can be specified by either the user (Native Mode) or by QuickControl (Normal Mode).



Native Mode

The example to the right shows the RSM command in Native Mode where the NVM Location is set to 1000 and registers 30 and 31 are being stored. In Native Mode, the user is responsible for keeping track of where data is being stored and making sure data does not overlap other data or programs.

Normal Mode

In Normal Mode, QuickControl automatically determines the NVM Location for all the NVM Commands at download time using the register files.

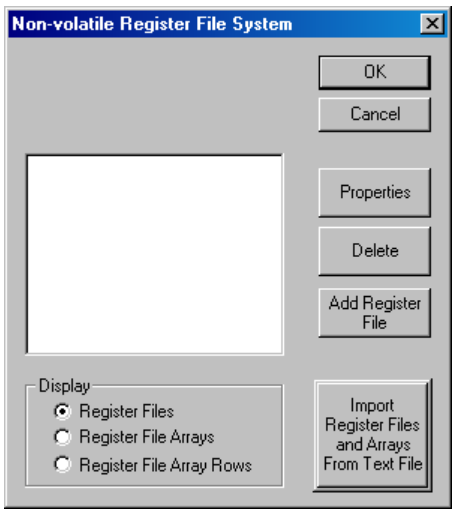
Register Files

A register file is a QuickControl object named by the user, which allows the user to specify the number of registers to store and their initial data.

Register files allow NVM Commands to refer to a register file instead of an explicit NVM Location. At download time, QuickControl automatically determines the best place in NVM to store the register data and copies these NVM Locations to the appropriate NVM Commands embedded in the user program.

Create a Register File

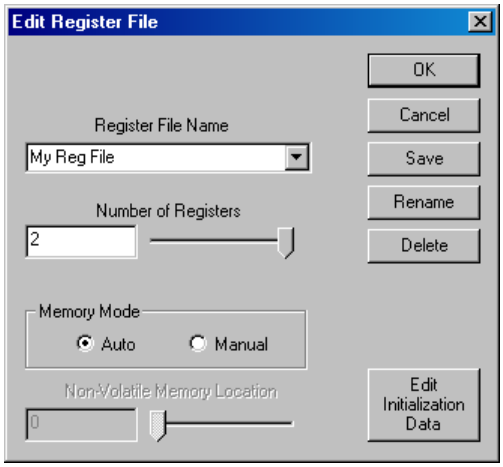
Register files are created from the menu Programs->Register Files.



The Non-Volatile Register File System dialog box displays a list of all register files. For a new program this list is blank.

To add a new
Register File, press
“Add Register File”.

Enter a unique name for the file and the number of registers to store. In the example to the right, "My Reg File" contains 2 registers.



The rest of the dialog box will be described latter.

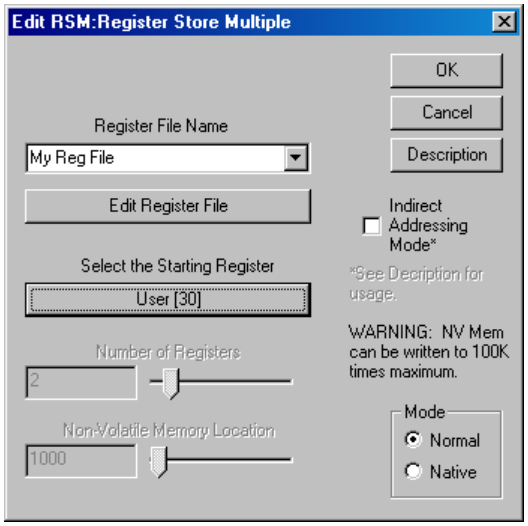
Press OK.

Your newly created register file will now appear on the list.

Press OK.

Using A Register File

With the register file “My Reg File” defined, the NVM Commands can now be used in Normal Mode without having to specify a NVM Location.



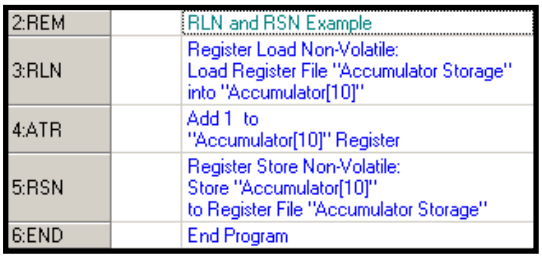
Let us use the RSM command with our new register file. The RSM command shown here uses the register file “My Reg File” to store registers 20 and 21 to NVM. Notice with Normal Mode selected, the NVM Location and Number of Registers controls are disabled. Remember, "My Reg File" contains this information so it does not need to be entered here by the user.

In this dialog box, the user can change which registers are stored and choose different register files.

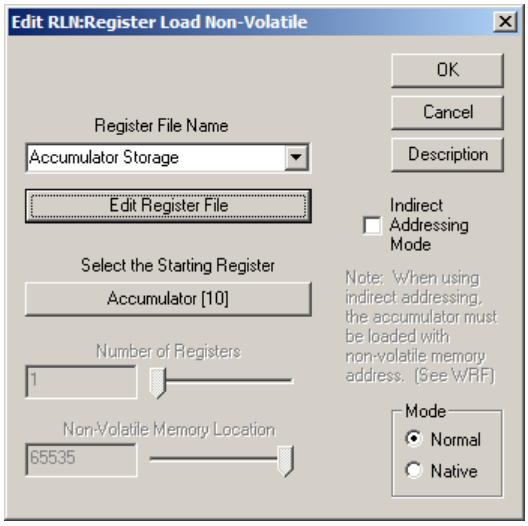
For use of Indirect Addressing, see Application Note “QCI-AN046 Indirect Addressing”.

Register Load and Store Example.qcp

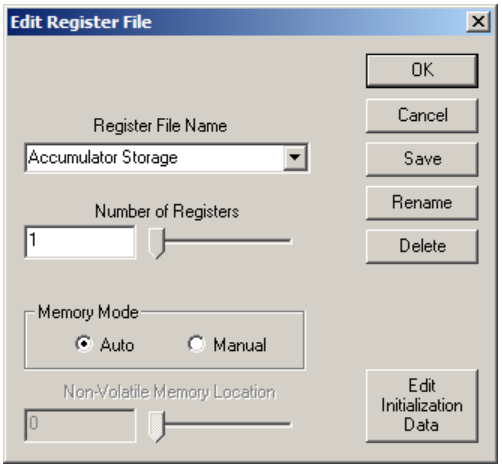
This example program demonstrates storing and loading data from a single register. The program loads register 10 from NVM, increments it, and then stores the data. If the Register Watch tool is used to monitor register Accumulator [10], you will note that it starts with a value of 10 and then increments every time power is cycled.



The register file “Accumulator Storage” properties can be edited by:
1) Double clicking on the RLN command.



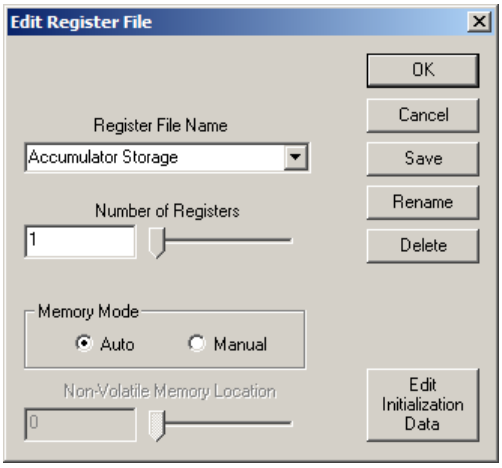
2) Press “Edit Register File”



Note “Number of Registers” is 1.

Why does register 10 start with a value of 10? See below.

Edit Register File Details



From the above example, the register file "Accumulator Storage" is shown on the right.

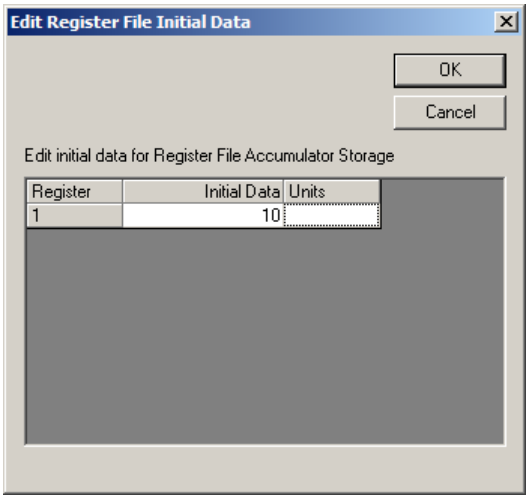
Register File Name
Name of register file. Must be unique within this Program File (QCP).

Number of Registers
Number of registers to store or load. Max 10.

Memory Mode
When Auto Memory Mode is selected (default), QuickControl assigns the register file's NVM Location during program download. Manual Memory Mode allows the user to specify the register file's NVM Location.

Non-Volatile Memory Location
The register file will be store at this location. Note, in Auto mode this is field is disabled.

Edit Initialization Data
At download time, QuickControl initializes the data at the register file's NVM Location to the value entered into the Edit Register File Initial Data dialog box. In the above example, press "Edit Initialization Data" to get the dialog box on the right. Note the Initial Data is 10. This is why the example program always resets to 10 when the program is downloaded.



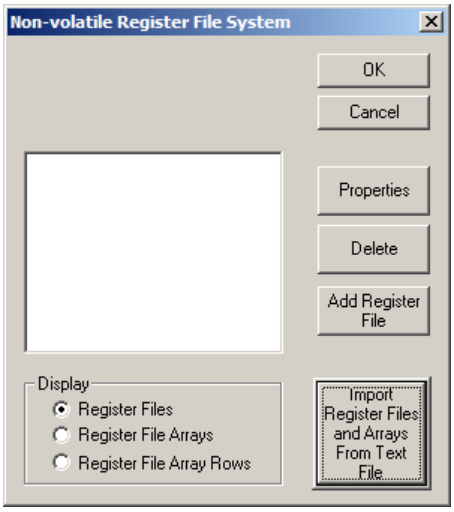
Double click on the Units cell to change the units.

Register File Array

A Register File Array is simply a collection of register files. Each entry (row) into the array is actually a register file. The number of registers (columns) for each row of the array must be the same. Register File Array creation is described below. Use of Register File Arrays is detailed in Application Note "QCI-AN046 Indirect Addressing".

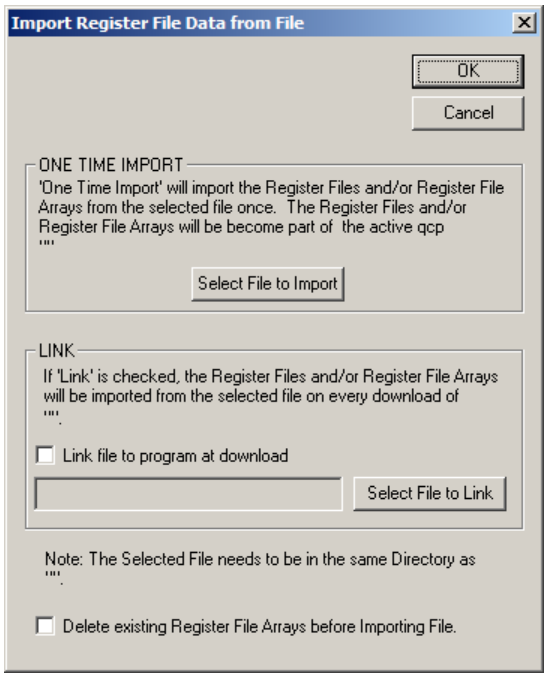
Import Register Files From Text File

Register files can be defined as detailed above or in a specially formatted text file (defined below). This “Import Text File” must be "linked" to the program file (qcp file) and saved in the same folder. The following procedure describes the linking of the Import Text File to a QCP:



From the Programs menu, click on register files to get the Non-Volatile Register File System dialog box as shown.

Press “Import Register Files and Arrays From Text File”.



You have the option of either doing a one time import or linking the Import Text file. Linking makes QuickControl import the text file each time the QCP is downloaded.

Note, the Import Text File needs to be in the same folder as the QCP.

Check “Delete existing….” to delete any previously imported Register File Arrays before importing the file. By default, importing will overwrite any objects with the same name, but will not delete any register files that are no longer defined in the text file.

NOTE: If the text file will import any data associated with units (see Data Format Directive), the QCP must have the appropriate scaling set and saved BEFORE the text file is linked.

Import Text File Format

Overview
"Reg File Adv Example.qcp" associated text files are a good example of Import Text Files.

An integral feature of the Register File System is its ability to dynamically link text files containing data to a QCP file that uses the data. These text files must reside in the same Windows folder as the QCP file that calls upon them. In addition, the text files must be formatted in such a way that allows QuickControl to read the data and reallocate it to non-volatile memory. This is achieved through Import Directives.

Comments
Any line preceded with a semi-colon is considered a comment and is ignored.
Example:
; Comment

Directives
The following formats are used to specify the Register File Import Directives.

File Include(@Include)
An @Include may be placed anywhere in the file and incorporates the listed file name as if it were part of this file.

Example: *@Include: Position Data File.txt*

Register File Import Directive @NVRegFile:
This directive informs QuickControl that a register file is to be imported.

Example: *@NVRegFile: startAdr=2500,col0Fmt=time*
There are two optional directives that may appear with the Register File Import directive, Starting Address and Data Format. If these directives are used, they need to be on the same line as the Register File Import Directive and must be separated by commas.

Starting Address Directive startAdr=
This directive specifies the non-volatile memory address of the register file. The starting address may be in decimal or hexadecimal form.

Example: *startAdr=2500*
If the starting address is not specified with this directive, the register file starting address will be set to the default Auto memory mode. In Auto memory mode, QuickControl assigns the register file starting address when the QCP is downloaded.

Data Format Directive colXFmt=
This directive specifies the format of imported data. When using this directive ensure that proper scaling is defined in the QCP file and saved BEFORE linking the text file to the QCP. The following data formats are accepted:

- *long* (imports data as a 32 bit signed value)
- *hex* (imports data as a hexadecimal value)
- *uLong* (imports data as a 32 bit unsigned value)
- *pos* (scales value and imports with position units set in QuickControl Scaling)
- *acc* (scales value and imports with acceleration units set in QuickControl Scaling)
- *vel* (scales value and imports with velocity units set in QuickControl Scaling)
- *time* (number of servo ticks [120 usec=1 tick])
- *torq* (percent torque)

The data format for each data entry (column) can be different. Putting the appropriate number in place of the X in the Data Format Directive specifies the data format of a specific data entry (column). Note the column numbers are zero-based (i.e. 0,1,2…), so that the first column's column number would be 0.

Example: *col0Fmt=time* (the format for the first column of data is time)
This data format directive will cause the first data entry of each register file that follows to be imported in time format. QuickControl will convert the time data to Native SilverLode Units. The data will be stored to memory in Native SilverLode Units.

Note: If the data format for an entry (column) is not specified with a directive, it defaults to long format.

Example: *@NVRegFile: startAdr=2500,col0Fmt=time*
(Using Register File Import Directive and both optional directives)
The register file data that follows this directive will be stored to non-volatile memory location 2500, and the first data entry (column) will be in time format.

Register File Details
The lines in the import text file following the Register File Directive should include details (name, # of registers, and data) for specific register files. There can be multiple register files imported, but the details of each register file must be contained on one line of the text file. The register file details must be separated by commas and listed in the following format:
<reg file name>, <# of registers>, <1st reg data>, <2nd reg data>, … <last reg data>

<reg file name> = Name of importing register file. Quotation marks are optional and spaces allowed.

<# of registers> = The number of registers to be stored in this register file. Range must be 1 to 10.

<1st reg data>, <2nd reg data>, … <last reg data> = The 32 bit data entries that are to be stored in this register file. The number of data entries should match the number of registers for this register file. Commas must separate the data entries.

Example: *Reg File 1, 2, 100,110*
This would define a register file named "Reg File 1" that has 2 registers with data 100 and 110.

Example: *@NVRegFile: startAdr=2500, col0Fmt=time, col2Fmt=pos, col4Fmt=vel*

Reg File 1, 2, 100, 110
Reg File 2, 6, 30,31,32,33,34,35

If an imported text file contained the above example, it would create two register files named "Reg File 1" and "Reg File 2". Reg File 1 would start at memory location 2500 and Reg File 2 would start at memory location 2506. After importing, non-volatile memory starting at address 2500 is be configured as follows:

Memory Address	# of words	Stored Elements
2500	1	5 (length - lower byte), 78 (checksum - upper byte)
2501	1	0
2502	2	833 (**time** data stored in native units)
2504	2	110
2506	1	13 (length - lower byte), 199 (checksum - upper byte)
2507	1	0
2508	2	250 (**time** data stored in native units)
2510	2	31
2512	2	32
2514	2	33
2516	2	273804 (**velocity** data stored in native units)
2518	2	35

Register File Details Alternative
An alternative to the format detailed above for importing Register File Details requires putting all of the information for the register file on the same line as the Register File Import Directive. The Register File Import Directive (@NVRegFile:) may be followed with the optional directives for starting address (startAdr=xxxx) and data format (colXFmt=format) as described above. In addition, the Register File Details may be supplied on the same line, using the following directives separated by commas.

Register File Name Directive name=
Specifies the name of the register file

Example: *name=reg file 1*

Number of registers Directive numRegs=
Specifies the number of registers in this register file and must be a number between 1 and 10.

Example: *numRegs=4*

Data Directive data=
This is an optional parameter if followed by data and must be the last directive. It specifies the 32-bit data to be stored to non-volatile memory. The data is to be enclosed in parenthesis and separated by commas.

Example: *data=(300,301,302,303)*

If a data directive is not given and the number of registers is defined, QuickControl will allocate memory for the register file, but will not save any data to memory.

Example: *@NVRegFile: name=reg file 1, numRegs=3*

If an imported text file contained the above example, it would create a register file named “reg file 1”. This register file would be assigned a non-volatile memory location by QuickControl, but that memory location would not be written to at program download.

The following examples use the above directives:

@NVRegFile: name=reg file 1, numRegs=3,startAdr=2800, col0Fmt=time, data=(300,301,302)
If an imported text file contained the above example, it would create a register file named “reg file 1”, which would be stored to non-volatile memory location 2800 as follows:

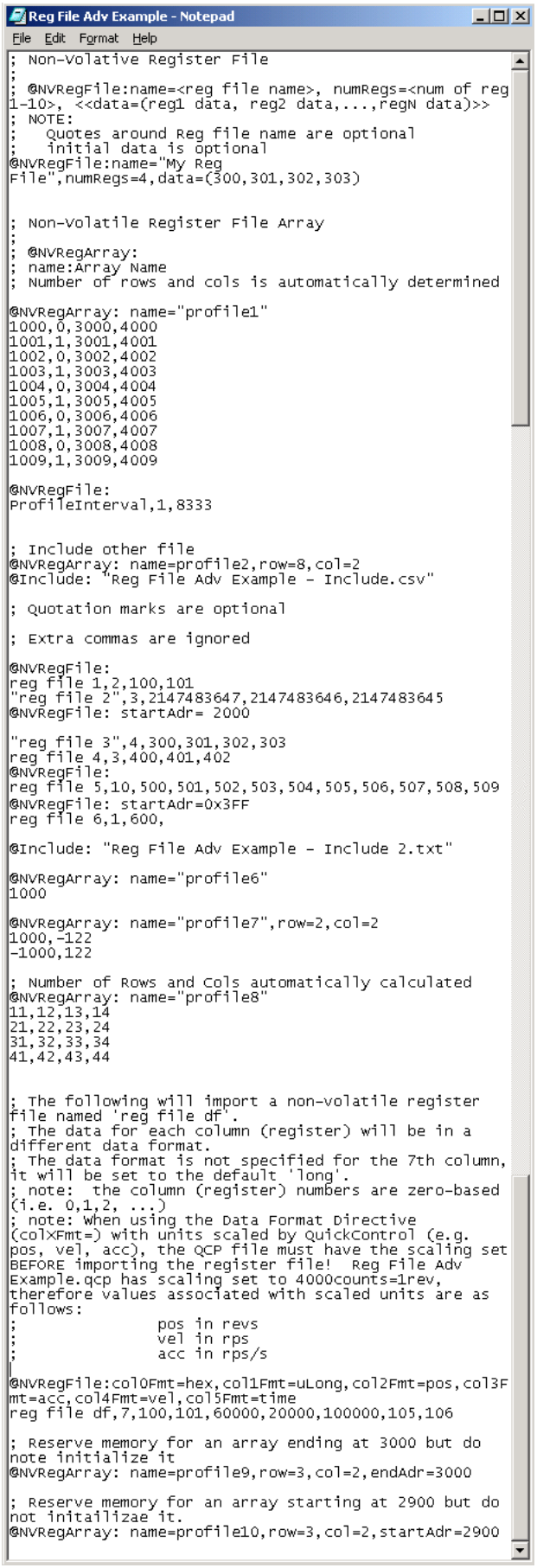
Memory Address	# of words	Stored Elements
2800	1	7 (length - lower byte), 215 (checksum -
2801	1	0
2803	2	2499 (**time** data stored in native units)
2805	2	301
2807	2	302

@NVRegFile: name=reg file 1, data=(300,301,302)
If an imported text file contained the above example, it would create a register file named “reg file 1”, stored to a non-volatile memory location assigned by QuickControl at program download.

Memory Address	# of words	Stored Elements
x	1	7 (length - lower byte), 118 (checksum -
x+1	1	0
x+3	2	300
x+5	2	301
x+7	2	302

Register File Array Import Directives

Import directives for Register File Arrays are very similar to those used with register files. Shown on the right is “Reg File Adv Example.txt”. This example is found in the “Data Registers” folder, within the “QCI Examples” directory and is used with “Reg File Adv Example.qcp”. The text file shown, illustrates the register file and Register File Array formatting techniques.



Register File Array Import Directive @NVRegArray:

This directive informs QuickControl that a Register File Array is to be imported.

The following directives define the Register File Array. They need to be on the same line as the Register File Array Import directive and must be separated by commas.

Register File Array Name Directive name=

Specifies the name of the Register File Array to be imported. Quotation marks optional and spaces allowed.

Example: *name=reg file array 1*

Number of Columns Directive col=

Specifies the number of columns in the Register File Array to be imported. The number of columns can be equated to the number of registers in each row of the Register File Array. Data must be between 1 and 10.

Example: *col=4*

This directive is optional unless using the ending address directive.

Number of Rows Directive row=

Specifies the number of rows in the Register File Array to be imported.

Example: *row=6*
This directive is optional unless using the ending address directive.

Starting Address Directive startAdr=
This directive specifies the non-volatile memory starting address of the Register File Array. The starting address may be in decimal or hexadecimal form.

Example: *startAdr=2500*

Ending Address Directive endAdr=
This directive specifies the non-volatile memory ending address of the Register File Array. The ending address may be in decimal or hexadecimal form. QuickControl will calculate the starting address for this Register File Array based on the given ending address. QuickControl needs to know the number of rows and columns to find the starting address, therefore the number of rows and columns must be specified before this directive.

Example: *endAdr=3000*

If neither the starting address nor the ending address is specified, the Register File Array starting address will be set to the default Auto memory mode. In Auto memory mode, QuickControl assigns the Register File Array starting address when the QCP is downloaded.

Data Format Directive colX**Fmt=**
This directive specifies the format of the imported data. When using this directive, ensure that proper scaling is defined in the QCP file and saved BEFORE linking the text file to the QCP. The following data formats are accepted:

- *long* (imports data as a 32 bit signed value)
- *hex* (imports data as a hexadecimal value)
- *uLong* (imports data as a 32 bit unsigned value)
- *pos* (scales value and imports with position units set in QuickControl Scaling)
- *acc* (scales value and imports with acceleration units set in QuickControl Scaling)
- *vel* (scales value and imports with velocity units set in QuickControl Scaling)
- *time* (number of servo ticks [120 usec=1 tick])

The data format for each column can be different. Putting the appropriate number in place of the X in the Data Format Directive specifies the data format of a specific column. Note the column numbers are zero-based (i.e. 0,1,2…), so that the first column's column number would be 0.

Example: ***col0Fmt=time*** (the first column's data format is **time**)
This data format directive will cause the first column of each Register File Array entry that follows to be imported in **time** format. QuickControl will convert the time data to Native SilverLode Units. The data will be stored to memory in Native SilverLode Units.

If the data format for a column is not specified with a directive, it will be set to the default **long** format.

Example: *@NVRegArray: name=reg file array 1, col=4, row=6, startAdr=2500, col0Fmt=time*

The Register File Array data that follows this directive will be stored to non-volatile memory location 2500, and the column will be in **time** format. There will be 6 entries in this array, and each entry has 4 columns.

Register File Array Details

The lines in the imported text file following the Register File Array Import Directive should include the 32-bit data for the Register File Array defined by the directive. The data of each Register File Array row must be contained on one line of the text file. The data must be separated by commas and listed in the following format:

<row 1: 1st reg data>, < row 1: 2nd reg data … < row 1: last reg data >
<row 2: 1st reg data>, < row 2: 2nd reg data … < row 2: last reg data >
…
<last row: 1st reg data>, <last row: 2nd reg data … < last row: last reg data >

If there is no Register File Array data and the rows and columns are defined, QuickControl will define the array and allocate memory for it, but will not write to memory.

Example: *@NVRegArray: name=reg file array 1, col=4, row=3, startAdr=2600, col0Fmt=time*

100,110,120,130
200,210,220,230
300,310,320,330

If an imported text file contained the above example, it would create a Register File Array named “reg file array 1”. The first row of this array would start at memory location 2600 followed in memory by the other rows of the array. After the import, non-volatile memory starting with address 2600 would be configured as follows:

Memory Address	# of words	Stored Elements
2600	1	9 length of reg file array 1 (first row) 84 checksum of reg file array 1(first row)
2601	1	0
2602	2	100 (**time** data stored in native units)
2604	2	110
2606	2	120
2608	2	130
2610	1	9 length of reg file array 1 (second row) 228 checksum of reg file array 1(second row)
2611	1	0
2612	2	200 (**time** data stored in native units)
2614	2	210
2616	2	220
2618	2	230
2620	1	9 length of reg file array 1 (third row) 113 checksum of reg file array 1(third row)
2621	1	0
2622	2	300 (**time** data stored in native units)
2624	2	310
2626	2	320
2628	2	330

Register File Storage Details

The first memory address of the register file always contains the Length and Checksum of the register file, which is automatically calculated and stored by the servo. The servo uses the length and checksum when loading the register file to know the correct number of words to load and to verify the accuracy of the data. The second word of the register file is a '0' or 'Null' and is added by the servo. The Null word is a safety feature to prevent the servo from trying to execute the data as a program. The data from the selected data registers is stored sequentially into the third and subsequent words of the register file.

Register File Memory Usage Example (starting at address 2500)

Memory Address	# of words	Stored Elements
2500	1	Length (lower byte) Checksum (upper byte)
2501	1	Null Word (0)
2503	2	Data from 1st Register
2505	2	Data from 2nd Register
2507	2	Data from 3rd Register
2509	2	Data from 4th Register
2511	2	Data from 5th Register
2513	2	Data from 6th Register
2515	2	Data from 7th Register

From the example, the total memory usage is 16 words to describe data in 7 registers.