Indirect Addressing

Associated QuickControl programs:
- Indirect Address – Native.qcp
- Indirect Address – NV Reg File.qcp

Overview

The SilverLode servos provide non-volatile (NV) memory storage for programs and data. Data can be stored as discrete elements or as part of arrays arranged in row(s) and column(s). Like all high-level programs, QuickControl uses indirect addressing to iterate through these array(s).

This application note describes SilverLode commands and QuickControl programs related to indirect addressing. The programs included show different indirect addressing methods. It is assumed the reader is familiar with QuickSilver's Register File System which is detailed in Application Note "QCI-AN048 Register Files".

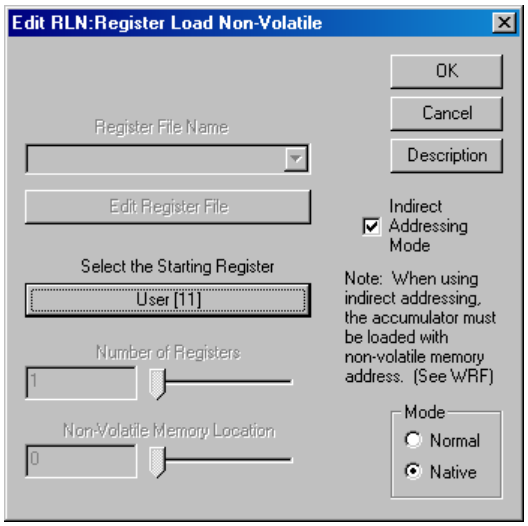
Indirect Address - Native.qcp

Program 1-A is the first half of the included program 'Indirect Address – Native.qcp' with all the comments removed. The program moves the servo specified by the values stored in non-volatile memory. You cannot write to NV memory directly. Data to be stored in NV memory must first be loaded into a Data Register as shown on lines 1 & 2 of Program 1-A. Use Write Register Program (WRP) to store data into Data Register (Accumulator[10]) then Register Store Non-Volatile (RSN) to store the value into NV memory. Program 1-A specifies NV memory location 1000, 1004, 1008 with 4000 counts, 8000 counts, and 12000 counts respectively.

Line# Oper	Label	Command
1:WRP		Write 4000 counts to "Accumulator[10]" Register
2:RSN		Register Store Non-Volatile: Store "Accumulator[10]" to Non-Volatile Memory Address 1000
3:WRP		Write 8000 counts to "Accumulator[10]" Register
4:RSN		Register Store Non-Volatile: Store "Accumulator[10]" to Non-Volatile Memory Address 1004
5:WRP		Write 12000 counts to "Accumulator[10]" Register
6:RSN		Register Store Non-Volatile: Store "Accumulator[10]" to Non-Volatile Memory Address 1008

Program 1-A

The second half of 'Indirect Address – Native.qcp' is shown in Program 1-B. Program 1-B shows how QuickControl handles indirect addressing.



This RLN command (line 8) was setup for indirect addressing and uses the Accumulator[10]'s value as a NV memory address.

The program does 3 passes to iterate through all three NV memory locations. Use Register Watch (Tools menu) and QuickControl's Single Step features to single step through the program (Indirect Address – Native.qcp) watching the registers at each step.

First Pass:

1. Accumulator[10] is loaded with 1000, the address of the first NV memory location.
2. RLN uses Accumulator[10] as an indirect reference into NV memory and loads the data (4000 counts) into register User[11].
3. RAT moves the servo 4000 counts at the specified ramp and total time.
4. Add To Register (ATR) adds 4 to the Accumulator[10], incrementing to the next NV memory location 1004.
5. Jump If Less Than (JLT) finds Accumulator[10] less than 1012 and jumps to LOOP.

Second Pass:

1. Accumulator[10] now contains 1004.
2. RLN uses Accumulator[10] as an indirect reference into NV memory and loads the data (8000 counts) into User[11].
3. RAT moves the servo 8000 counts.
4. ATR adds 4 to the Accumulator[10], incrementing to the next NV memory location 1008.
5. JLT finds Accumulator[10] less than 1012 and jumps to LOOP.

7:WRP		Write 1000 to "Accumulator[10]" Register
8:RLN	LOOP	Register Load Non-Volatile: Load "User[11]" from NV Address in "Accumulator[10]"
9:RAT		Move to location stored in "User[11]" Register @ ramp time=100 mSec total time=1000 mSec
10:ATR		Add 4 to "Accumulator[10]" Register
11:JLT		Jump to "LOOP" When "Accumulator[10]" < 1012
12:END		End Program

Program 1-B

Third Pass:

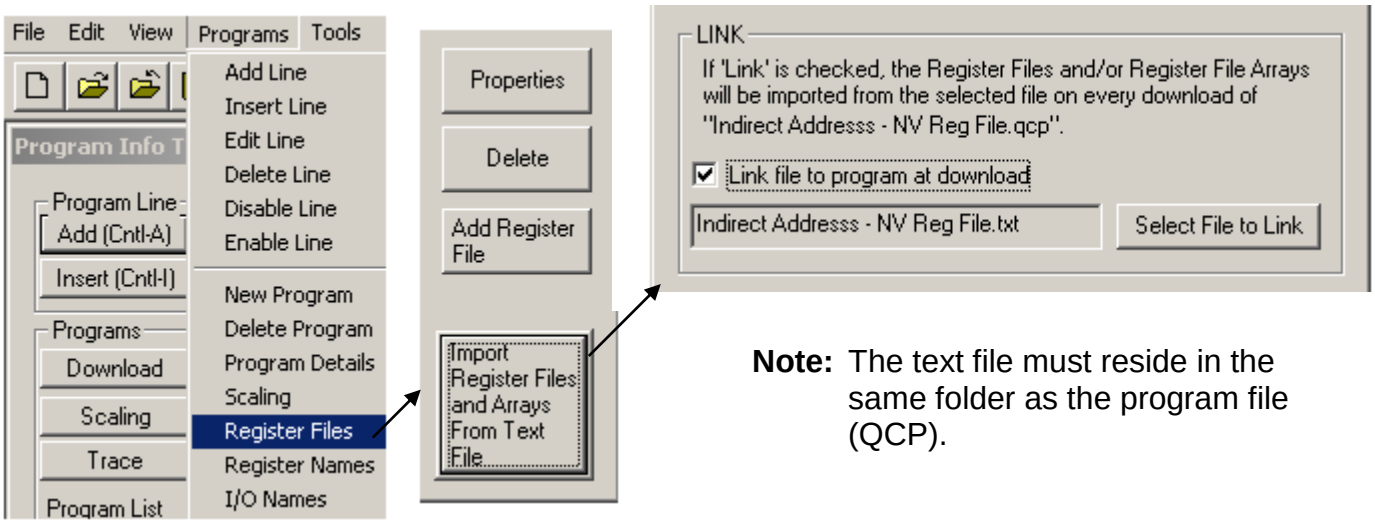
1. Accumulator[10] now contains 1008.
2. RLN uses Accumulator[10] as an indirect reference into NV memory and loads the data (12000 counts) into User[11].
3. RAT moves the servo 12000 counts.
4. ATR adds 4 to the Accumulator[10], incrementing to the next NV memory location 1012.
5. JLT finds Accumulator[10] not less than 1012 and continues on to the next line (End Program).

Indirect Address - NV Reg File.qcp

The QuickControl program file 'Indirect Address - NV Reg File.qcp' uses registers[12-14] to iterate through a Register File Array. Note, registers[12-14] have been renamed for this example. See Application Note "QCI-AN048 Register Files" for more details.

NOTE: If a SilverDust is being used, see Indirect Address - NV Reg File SD.qcp below for an alternative method.

'Indirect Address - NV Reg File.qcp' is linked to a text file named 'Indirect Address – NV Reg File.txt' – Diagram 1. The text file specifies a Register File Array named 'moves' with an array of three rows and one column – Diagram 2. The program file (QCP) has been setup to automatically import the text file every time the QCP is downloaded (see Diagram 1). For this example, QuickControl controls memory management for efficient use of memory space. Upon import, QuickControl allocates memory space accordingly, names the array 'moves' and inserts the data.



Note: The text file must reside in the same folder as the program file (QCP).

Diagram 1

Note: Text file must have a proper heading as shown in Diagram 2. See QCI-AN048 Register Files regarding register file format convention.

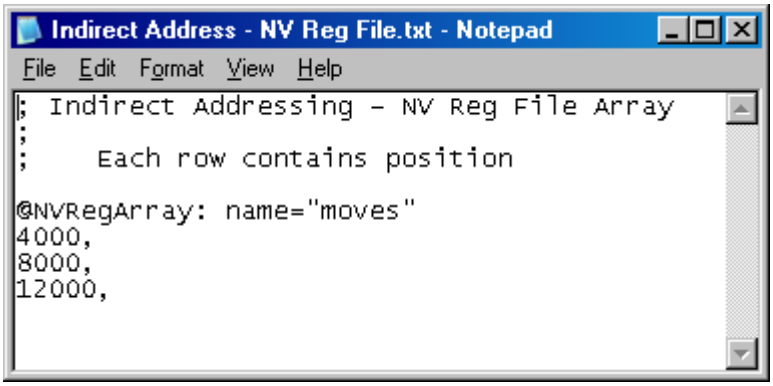


Diagram 2

With an external file imported, WRF will create a link to the Register File Array ‘moves’ and obtain its properties – Diagram 3. The properties are loaded into registers[12-14] one at a time (see below). The information is used to iterate through the Register File Array.

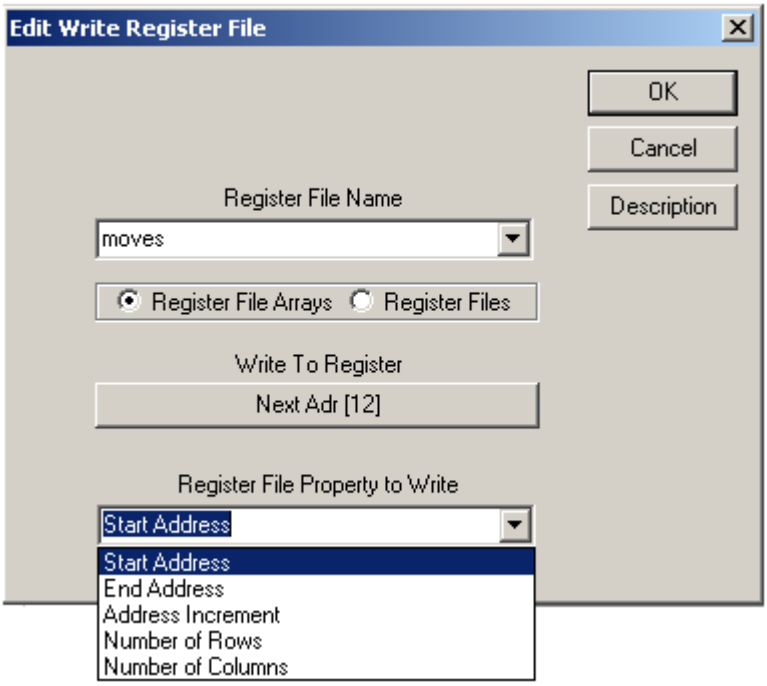


Diagram 3

Program 2-A is a portion of the included ‘Indirect Addressing – NV Reg File.qcp’ program file with all comments removed. The program starts with WRF commands writing the array's properties Starting Address, Address Increment, and Number of Rows to registers[12-14] respectively. Accumulator[10] is loaded with the Starting Address (register 12). RLN loads the data from NV memory into User[11]. Recall that RLN automatically uses the Accumulator[10] as an indirect address to a NV memory location.

Line# Oper	Label	Command
1:WRF		Write "Start Address" of Register File Array "moves" to "Next Adr[12]"
2:WRF		Write "Address Increment" of Register File Array "moves" to "Adr Increment[13]"
3:WRF		Write "Number of Rows" of Register File Array "moves" to "Num Rows[14]"
4:CLC		"Accumulator[10]" = "Next Adr[12]"
5:RLN	GET DATA	Register Load Non-Volatile: Load "User[11]" from NV Address in "Accumulator[10]"
6:RAT		Move to location stored in "User[11]" Register @ ramp time=100 mSec total time=1000 mSec

Program 2-A

Example 1 shows how the Register File Array ‘moves’ would be stored in NV memory. X represents the NV memory address of the array's data. QuickControl determines X unless manually defined by the user. In this example, the Address Increment is 2 and Number of Rows is 3.

Example 1:

@NVRegArray: name="moves",row=3,col=1
4000,
8000,
12000,

Memory Address	# of words	Stored Elements
x	1	(Length – Lower byte) (Checksum – Upper byte)
X+1	1	0
X+2	2	4000
X+4	2	8000
X+6	2	12000

Program 2-B shows the rest of ‘Indirect Address – NV Reg File.qcp’. The program will make three passes. Use Register Watch (Tools menu) and QuickControl's Single Step features to single step through the program watching the registers at each step.

First Pass:

1. Accumulator[10] is loaded with reg[12] – the start of the Register File Array ‘moves’.
2. RLN uses the address in Accumulator[10] to load row 1’s data (4000 counts) into User[11].
3. RAT moves the servo 4000 counts.
4. Register [13] (Address Increment) is added to register[12] to equal row 2's starting address – lines 7, 8 and 9.
5. Register[14] decrements to two rows remaining – line 10.
6. JMP checks for remaining row(s) and loops back to GET DATA. Two rows remaining.

4:CLC		"Accumulator[10]" = "Next Adr[12]"
5:RLN	GET DATA	Register Load Non-Volatile: Load "User[11]" from NV Address in "Accumulator[10]"
6:RAT		Move to location stored in "User[11]" Register @ ramp time=100 mSec total time=1000 mSec
7:CLC		"Accumulator[10]" = "Next Adr[12]"
8:CLC		"Accumulator[10]" = "Accumulator[10]" + "Adr Increment[13]"
9:CLC		"Next Adr[12]" = "Accumulator[10]"
10:CLC		Decrement "Num Rows[14]"
11:JMP		Jump to "GET DATA" If Last Calc was Positive
12:END		End Program

Program 2-B

Second Pass:

1. Accumulator[10] now contains row 2's NV memory address.
2. RLN uses the address in Accumulator[10] to load row 2's data (8000 counts) into User[11].
3. RAT moves the servo 8000 counts.
4. Register [13] (Address Increment) is added to register[12] to equal row 3's starting address – lines 7, 8 and 9.
5. Register[14] decrements to one row remaining – line 10.
6. Jump (JMP) checks for remaining row(s) and loops back to GET DATA. One row remaining.

Third Pass:

1. Accumulator[10] now contains row 3's NV memory address.
2. RLN uses the address in Accumulator[10] to load row 3's data (12000 counts) into User[11].
3. RAT moves the servo 12000 counts.
4. Register [13] (Address Increment) is added to register[12] to equal the array's ending address - lines 7, 8 and 9.
5. Register[14] decrements to zero rows remaining – line 10.
6. Jump (JMP) finds no rows remaining and continues to the next line (End Program).

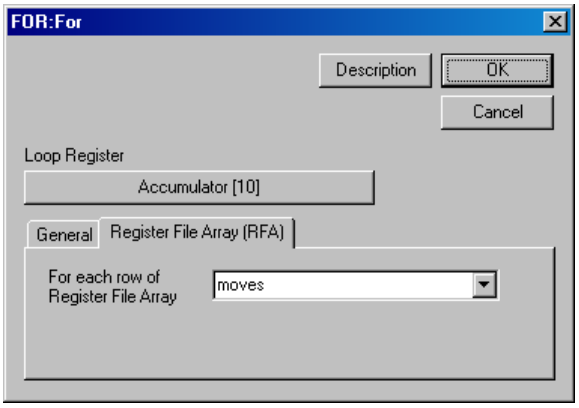
Indirect Address - NV Reg File SD.qcp

The QuickControl program file 'Indirect Address - NV Reg File SD.qcp' uses the SilverDust commands FOR and NXT to greatly simplify the process of iterating through a Register File Array.

NOTE: SilverDust Rev 28 and QuickControl Rev 4.5 are required.

The program loads the same array as "Indirect Address - NV Reg File.qcp" and performs the same moves, but the SilverDust FOR command automatically increments the Accumulator from the starting address of the first row to the starting address of the last row.

3:REM		This FOR command loads the Accumulator with the starting address of the next row in the "moves" array.
4:FOR		FOR "Accumulator[10]" = each row in Register File Array moves
5:RLN		Register Load Non-Volatile: Load "User[11]" from NV Address in "Accumulator[10]"
6:RAT		Move to location stored in "User[11]" Register @ ramp time=100 mSec total time=1000 mSec
7:NXT		Next (FOR line 4)
8:END		End Program



The FOR command only needs the Register File Array name and what register to put the starting address in.

If a SilverDust is being used, this is the method of choice.