Camming

The following files are used in this application:

CamSN-RSD.qcp:	Uses RSD to implement cam table. Use with SilverNugget.
CamSD-EGMTrap.qcp:	Uses EGM to implement cam table. Use with SilverDust.
CamTable.txt:	Text file of cam table used by QuickControl for storing data into NVM.
CamTable.xls	Spreadsheet of cam table with the graph of cam data.

This application note describes how a SilverLode servo can follow a Cam Profile defined by a data table stored into non-volatile memory (NVM). The servo tracks the speed and position of a master encoder source while moving to the specific positions designated by the cam table.

Prerequisites

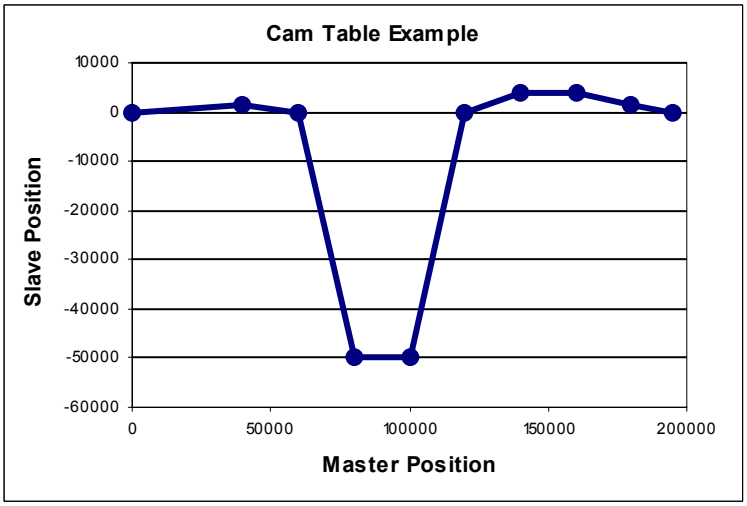
It is assumed that the user is already familiar with the following:
1 - Electronic Gearing (See App Note QCI-AN019 Electronic Gearing)
2 - Indirect Addressing (see App Note QCI-AN046 Indirect Addressing)
3 - Register File Arrays (see App Note QCI-AN048 Register Files)

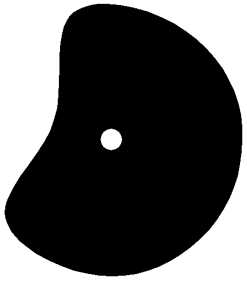
Camming Theory

Camming is a servo's equivalent of a mechanical cam attached to some sort of input. A typical cam is an irregularly shaped disc, driven by a shaft, with a sensor (such as an idle wheel or dancer arm) riding along its circumference. The shape of the cam determines the position of the dancer (generally called the “slave”) relative to the position of the shaft (or “master”). In the case of the disc/idler arm combination, rotary position of the disk determines the linear displacement of the arm. Generically, the position (rotary, linear, or other) of the input determines the position of the output. A smooth camming setup has the added benefit of adjusting the output velocity to move smoothly between the points as the master moves along.

A cam table is a text representation of the master/slave relationship. It consists of many rows and two columns. The first column is the master position and the second column the corresponding slave position. Referring back to the disc/arm example, the master position is the rotary location, and the slave location the radius of the disk at that location. In order for the slave to correctly follow the cam table, it must reach the slave position at the same time the master reaches the master position in the each row.

Row	Master Position	Slave Position
1	0	0
2	40000	1500
3	60000	0
4	80000	-50000
5	100000	-50000
6	120000	0
7	140000	4000
8	160000	4000
9	180000	1350
10	195000	0
10	200000	0





Above is the cam example used in this application note. A cam table, the corresponding graph of the cam data, and an image of the physical cam is provided.

Definitions
Current Master Position (M)
Next Master Position (Mn)
Current Slave Position (S)
Next Slave Position (Sn)
Change in Slave Position (CS) = Mn-M
Change in Master Position (CM) = Sn-S

QuickSilver provides two ways of programming a cam table. For SilverDust controllers, the recommended way is using the Electronic Gearing Mode (EGM) Trapezoid Move feature. For SilverNugget controllers use the Registered Step & Direction (RSD) command. See examples below for more details.

CamSD-EGMTrap.qcp

This example uses the Electronic Gearing Mode (EGM) Trapezoid Move feature to follow the cam table. It requires

- SilverDust Rev 37
- QuickControl Rev 4.64

First part of the program configures the external encoder and stops any motion.

For Each Row….

The cam table is stored in NV Memory as a 2 column Register File Array (see App Note QCI-AN048 Register Files). The FOR/NXT commands automatically iterate through the table (App Note QCI-AN046 Indirect Addressing).

We keep jumping back to LOOP until the External Encoder Position passes Mn signifying it is time for the next row of data which is done in the subroutine GET NEXT ROW.

Line# Oper	Label	Command
16:REM		========== For Each Row in Cam Table ==========
17:FOR		FOR "Next Adr[12]" = each row in Register File Array "Cam Table"
18:REM	LOOP	Wait for Next Cam Table Row Loop
19:REM		Add other application specific code here
20:REM		Keep looping if Mn>M
21:CLX		Accumulator[10] = Mn[15] - External Encoder Position[200]
22:JMP		Jump to "LOOP" If Last Calc Was Positive TRUE
23:REM		Get Next Cam Table Row
24:PCL		Program Call "GET NEXT ROW"
25:REM		Next
26:NXT		Next (FOR line 17)
27:REM		Re-initialize Cam table and start over again
28:JMP		Jump to "START"

Line# Oper	Label	Command
29:REM	GET NEXT ROW	========== Subroutine: Get Next Cam Table Row ==========
30:REM		Current Master Position (M) Next Master Position (Mn) Current Slave Position (S) Next Slave Position (Sn) Change in Slave Position (CS) = Mn-M Change in Master Position (CM) = Sn-S
31:REM		Read the next row of data (Mn,Sn) from the Cam Table stored in non-volatile memory. NOTE: Indirect addressing is being used. See app note for details.
32:CLC		Accumulator[10] = Next Adr[12]
33:RLM		Register Load Multiple: Load 2 Registers starting w/ "Mn[15]" from NV Address in Accumulator(10)
34:REM		CS = (Sn-S)
35:CLX		CS[27] = Sn[16] - Target Position[0]
36:REM		CM=(Mn-M)
37:CLX		CM[26] = Mn[15] - External Encoder Position[200]

Subroutine: GET NEXT ROW

We copy the next row NV Mem address into the Accumulator for the RLM command to use to load the next row (Mn and Sn).

Calculate Change in Slave (CS) and Change in Master (CM).

Configure the EGM command to move the servo CS counts as the master moves CM counts.

The EGM command just does a Trap Move for each row the table.

Line# Oper	Label	Command
38:REM		EGM Trapezoid Move Master Distance (MD)=CM Master Ramp Distance (MRD) = ABS(MD/10) Trap Move Distance (TMD) = CS Set SF=0 because there is no "base" gear ratio
39:WRP		Write 0 to "SF[51]" Register
40:WRP		Write 0 to "AF[50]" Register
41:WRP		Write 0 to "SP[52]" Register
42:CLX		MD[53] = CM[26]
43:CLD		MRD[54] = MD[53]/LO(10)
44:CLC		MRD[54] = ABS(MRD[54])
45:CLX		TMD[55] = CS[27]
46:REM		Skip if TMD or MD == 0
47:JRE		Jump 3 line(s) When "TMD[55]" = 0
48:JRE		Jump 2 line(s) When "MD[53]" = 0
49:EGM		Electronic Gearing Mode: Acc Factor in "AF[50]" Scale Factor in "SF[51]" Trapezoid Move Enabled Start Position (SP) in Register 52 Master Distance (MD) in Register 53 Master Ramp Distance (MRD) in Register 54 Trapezoid Move Distance (TMD) in Register 55 Modulo Master Distance (MMD) in Register 56 Override Existing Move Relative Master Position One Trapezoid Move Only
50:PRT		Program Return:

Cam Profile SN - RSD.qcp

The SilverNugget does not have the EGM command, so it must calculate the required Gear Ratio (GR) to make the slave move the required distance.

The gear ratio (GR) determines the relationship of the following servo motor (or slave) to the master encoder. To be more specific, GR is defined as the slope of the cam table's active segment.

Slave Position = GR * Master Position

GR is defined as: GR = CS/CM = (Sn-S)/(Mn-S)

For example:
When the master reaches 40,000 (M) (row 2) its next position is 60,000 (Mn). The current position of the slave would then be 1,500 (S) with its next position being 0 (Sn)

GR = (Sn-S)/(Mn-M)
GR = (0-1,500)/(60,000-40,000)
GR = -0.075

Therefore; Slave Position = GR * Master Position

In the above GR example, the slave would reach 0 the same time as the master reached 60,000. GR must be recalculated each time the master reaches the next row in the table.

First part of the program configures the external encoder and stops any motion.

Start Camming

The SilverNugget does not support the FOR/NXT commands so iterating through the cam table requires more instructions. Notice the WRF commands are used to setup some registers for indirect addressing. The RSD command is then executed in the background with a Scale Factor (SF) = 1:1 gear ratio.

Line# Oper	Label	Command
7:REM	START	========== Start Camming ==========
8:REM		Stop any motion
9:DMT		Disable Multi-Tasking
10:VMP		Velocity Mode: acc=40000 cps/s, vel=0 cps
11:REM		Clear variables
12:CLC		Clear External Encoder Position[200]
13:CLC		Clear Mn[15]
14:REM		Setup Indirect Address registers
15:WRF		Write "Start Address" of Register File Array "Cam Table" to "Next Adr[12]"
16:WRF		Write "Address Increment" of Register File Array "Cam Table" to "Adr Increment[13]"
17:WRF		Write "Number of Rows" of Register File Array "Cam Table" to "Num Rows[14]"
18:REM		We need multi-tasking
19:EMT		Enable Multi-Tasking
20:REM		RSD Scale Factor (SF) Give a 1:1 value to start with
21:CLC		Accumulator[10] = SF1[29]
22:CLC		SF[28] = Accumulator[10]
23:RSD		Registered Step and Direction: Scale Factor = "SF[28]" Register

Wait for Next Cam Table Row Loop
We keep jumping back to LOOP until the External Encoder Position passes Mn signifying it is time for the next row of data which is done in the subroutine GET NEXT ROW.

Line# Oper	Label	Command
24:REM	LOOP	Wait for Next Cam Table Row Loop
25:REM		Add other application specific code here
26:REM		Keep looping if Mn>M
27:CLC		Accumulator[10] = Mn[15]
28:CLC		Accumulator[10] = Accumulator[10] - External Encoder Position[200]
29:JMP		Jump to "LOOP" If Last Calc Was Positive TRUE
30:REM		Get Next Cam Table Row
31:PCL		Program Call "GET NEXT ROW"
32:REM		Calc NV Mem adr for next row
33:CLC		Accumulator[10] = Next Adr[12]
34:CLC		Accumulator[10] = Accumulator[10] + Adr Increment[13]
35:CLC		Next Adr[12] = Accumulator[10]
36:CLC		Decrement Num Rows[14]
37:JGE		Jump to "LOOP" When "Num Rows[14]" >= 0
38:REM		Re-initialize Cam table and start over again
39:JMP		Jump to "START"

Subroutine: GET NEXT ROW
We copy the next row NV Mem address into the Accumulator for the RLM command to use to load the next row (Mn and Sn).

Line# Oper	Label	Command
40:REM	GET NEXT ROW	========== Subroutine: Get Next Cam Table Row ==========
41:REM		Current Master Position (M) Next Master Position (Mn) Current Slave Position (S) Next Slave Position (Sn) Change in Slave Position (CS) = Mn-M Change in Master Position (CM) = Sn-S
42:REM		Read the next row of data (Mn,Sn) from the Cam Table stored in non-volatile memory. NOTE: Indirect addressing is being used. See app note for details.
43:CLC		Accumulator[10] = Next Adr[12]
44:RLM		Register Load Multiple: Load 2 Registers starting w/ "Mn[15]" from NV Address in Accumulator[10]
45:REM		CS = (Sn-S)
46:CLC		Accumulator[10] = Sn[16]
47:CLC		Accumulator[10] = Accumulator[10] - Actual Position[1]
48:CLC		CS[27] = Accumulator[10]
49:REM		CM=(Mn-M)
50:CLC		Accumulator[10] = Mn[15]
51:CLC		Accumulator[10] = Accumulator[10] - External Encoder Position[200]
52:CLC		CM[26] = Accumulator[10]

Calculate Change in Slave (CS) and Change in Master (CM).

Calculate Scale Factor (SF)
SF = SF1 * GR
Where SF1 = Scale Factor 1:1 Ratio

SF1 is encoder dependent and is defined as follows:

Encoder CPR	Scale Factor 1:1 Ratio (SF1)
4000	1024
8000	512
16000	256

GR = CS/CM

Therefore

SF = SF1 * CS/CM

Line# Oper	Label	Command
53:REM		Scale Factor 1:1 Ration (SF1): SF value for a 1 to 1 ration. SF1 is encoder CPR dependent: 4000CPR; SF1 = 1024 8000CPR; SF1 = 512 16000CPR; SF1 = 256 Change this value for your encoder CPR
54:WRP		Write 1024 to "SF1[29]" Register
55:REM		Calculate Scale Factor (SF) SF = SF1 * GR Scale Factor For 1:1 Ratio (SF1) - see below Gear Ratio (GR) GR = CS/CM Therefore SF = SF1 * CS/CM
56:CLC		Accumulator[10] = SF1[29]
57:CLC		Accumulator[10] = Accumulator[10] * CS[27]
58:CLC		Accumulator[10] = Accumulator[10]/CM[26]
59:CLC		SF[28] = Accumulator[10]
60:PRT		Program Return:

Tuning Parameters
RSD has tuning parameters typically adjusted as follows:

Control Constants (CTC)	
Kp	Reduce default by 50%
Kv1	0
Kv2	Reduce default by 25%
Kvff	Same as Kv2
Ka	0
Kaff	0
Ki	Between 50 and 100

Filter Constants (FLC)	
Fv1	30
Fv2	90
Fa	Default

These values are recommended default values. Specific applications should be tuned using these parameters as a starting point.

Warnings and Recommendations for Best Results

1) Keep the table as smooth as possible by limiting how much the gear ratio changes from one row to the next. The largest errors will occur during dramatic changes in the gear ratio.
2) Master positions do not have to be evenly spaced. Take advantage of this by putting more points around any required sharp transitions. See Minimum Spacing below for limitations.
3) Keep the slave velocity less than three times the master velocity. Larger gear ratios can cause jittering. This is the side effect of the relatively few pulses received from the master. Inputs from higher resolution encoders (and the corresponding lower gear ratio) will improve performance.
4) Each row of cam data requires 6 words of non-volatile memory. The data is automatically stored from the top of memory down, but will eventually run into the application code if the data size gets too large. QuickControl will provide an error at download if this occurs.
5) Minimum Spacing and High Speed: In the example programs the calculation times are approximately 5ms. For best results, keep the total time between rows, which is dependent on master speed and cam table point spacing, to be at least 5 times this calculation time.
6) In “CamSN-RSD.qcp”, Scale Factor (SF) is an integer. Small changes in slave position verses large changes in master position (i.e. 1:100) will result in poor performance due to SF having a small dynamic range (i.e. -5 to +5).

Changing Direction

The example provided with this application note assumes unidirectional operation. More complex programming is required to allow the following servo motor to properly track a master encoder moving in either direction.