EUROTHERM
DRIVES

Servo Application Manual

Cut To Length

Prepared by: Phil Suffolk

Date: 16th Nov 1999

Distribution List: EDL Service Engineers

Contents

1.0 Machine Overview

Fig 1.0 illustrates the typical arrangement of a cut to length machine.

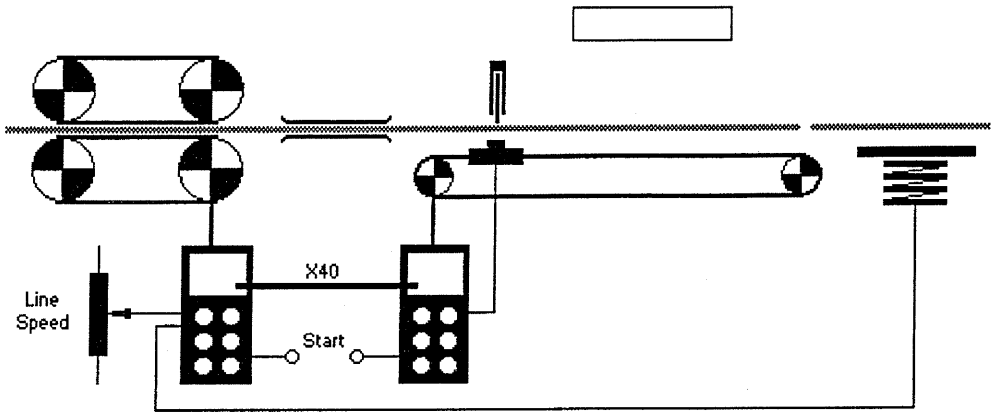

Fig 1.0 – Standard cut to length configuration.

The haul-off will provide a line encoder signal either directly or from the emulated encoder output. The number of pulses would be proportional to the length of the material passed. On detecting the correct number of pulses the cutter ramps to synchronous speed, provides a digital output to activate the blade and then ramps down. On completion of the cycle the cutter is returned to it's home position.

Some form of material handler is usually included to remove the processed material from the process. A signal from the haul-off may provide a suitable 'eject' signal.

1.1 Machine Limitations

Machine cut tolerance will be a combination of two main factors.

- Accuracy of line encoder. Material contact problems or mechanical imperfections may both lead to an encoder count that is not proportional to exact material length.
- Servos scan error. The servo scans for target encoder counts every 2msec. The length of material, which passes through the haul off in this time frame, generates a cut error.
- Cut quality. Movement in the blade mechanics will generate cut length inaccuracies.

The minimum machine cycle time provides the lower limit for the cut length. Machine inertia dictates the rate at which the cutter can travel through its synchronous move phase and then return to home. The torque rating of the motor should be calculated with this in mind.

2.0 Control Wiring Overview

Fig 2.0 illustrates a typical wiring arrangement for implementing a cut to length system using a 631. The line encoder would be connected to the drive's X40 socket and is not illustrated.

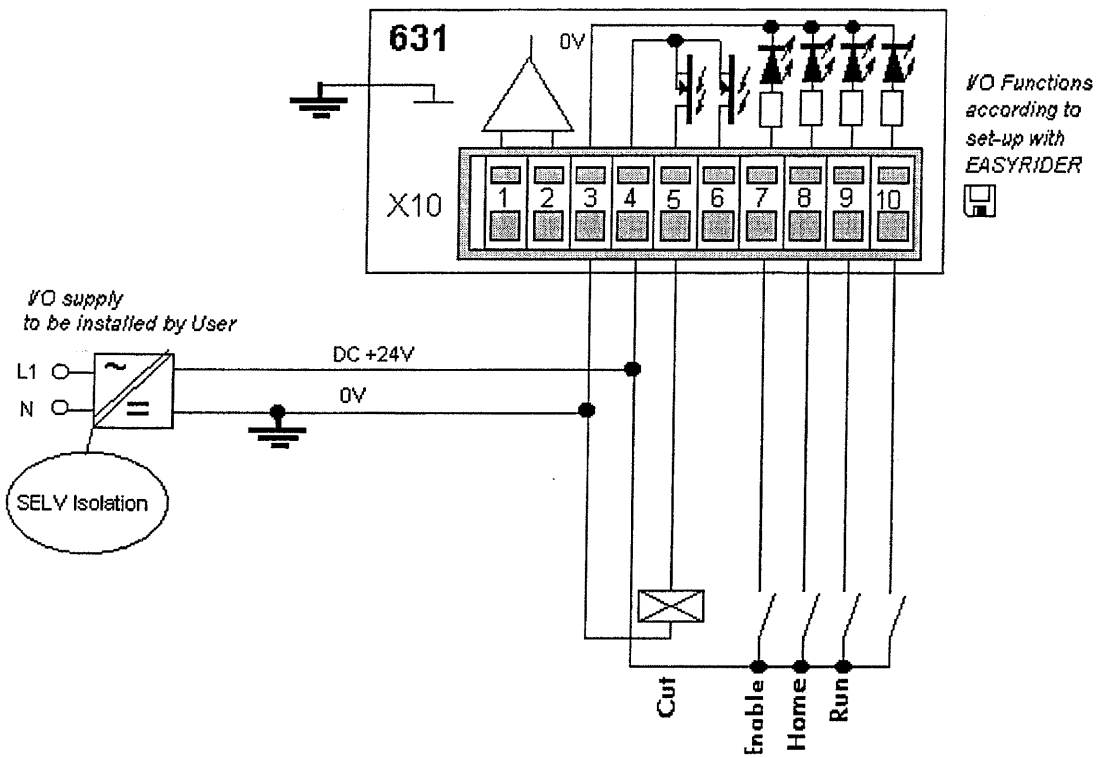

Fig 2.0 – Suggested control wiring

3.0 Control Philosophy

Operating the 'enable' input will activate the drive and start the BIAS program. The drive will move the cutter until it detects the 'home' input. Once at home the resolver and reference counts are reset.

The drive waits for a 'run' signal and then performs it's first cut cycle. The reference encoder count is then monitored and the cut cycle triggered after the perquisite number of counts have been detected.

The cut cycle is repeated until the 'run' or 'enable' signal has been removed.

4.0 Example BIAS Program

Fig 3.0 lists a typical BIAS program for achieving the cut cycle for the cut to length application.

```
PROG_START:
0     act. posit. 1 = 0              INKR
1     act. posit. 2 = 0              INKR
2     [variable 0  ] = 0
3     [variable 1  ] = 0
4     acceleration    = 50000  rpm/s
5     deceleration    = 50000  rpm/s
6     gear factor  = 1
7     synch.adjustment;     linear= 1 ,mode = 0    , val.= 1
8     synchr.settings;mode =130  offset=[var.1  ]; startoffset=[var.1  ]
9     [variable 0  ] = 100000
START_NEXT:
10    position =  16384        INKR
11    start axis
12    move synchron
13    position =  32768        INKR
14    update parameter
WAIT1:
15    If   status 2   == 0 then jump     WAIT1
16    output  20  = 1
17    position =  16384        INKR
18    update parameter
19    synchr.settings;mode =1  ; offset=[var.1  ]; startoffset=[var.1  ]
WAIT2:
20    If   status 2   == 0 then jump     WAIT2
21    output  20  = 0
WAIT3:
22    If   status 2   == 0 then jump     WAIT3
23    wait time    100    ms
24    start axis
25    move position;         v = 3000  rpm,  s= 0            INKR
26    wait for position reached
WAIT_MASTR:
27    If   actual pos. 2 <  [variable 0  ] then jump     WAIT_MASTR
28    synchr.settings;mode =130  offset=[var.1  ]; startoffset=[var.0  ]
29    [variable 0  ] = [variable 0  ] + 100000
30    jump      START_NEXT
```

Fig 4.0 – Example BIAS code for cut cycle only

The principle settings are programmed using the 'synchr.settings...' commands. A summary of the parameters can be found in Appendix A.

Fig 4.1 illustrates the operation of the above code as a function of time.

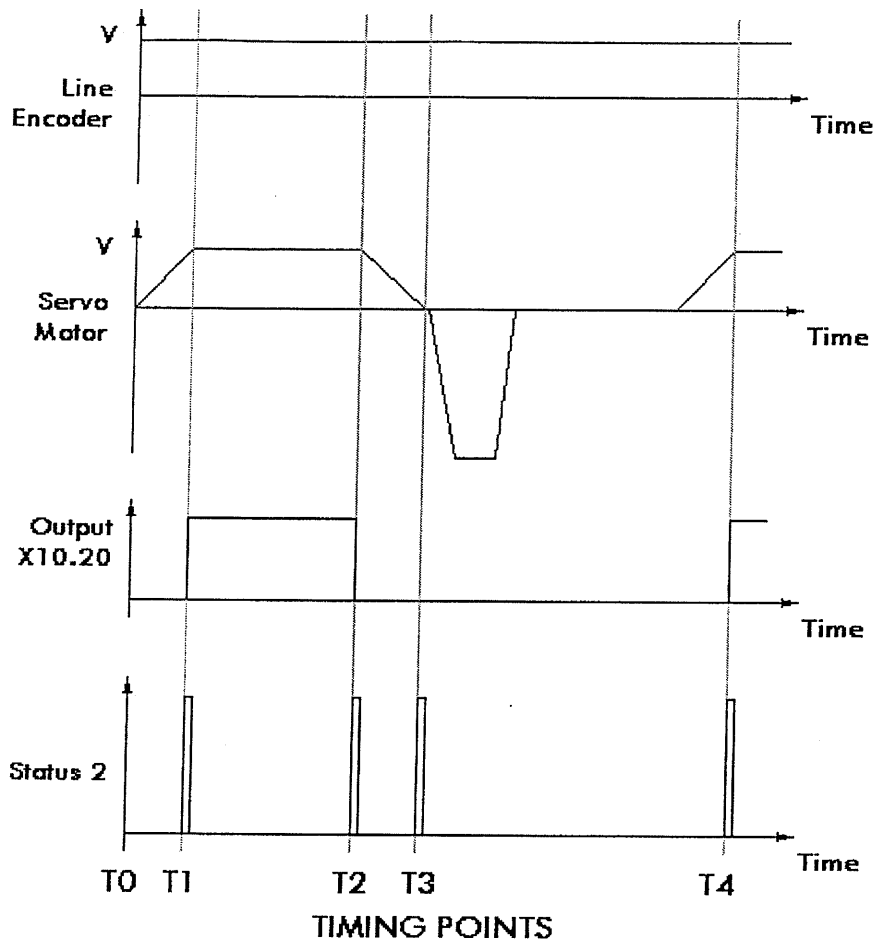

Fig 4.1 – Process Cycle Timing Information

The relationship between line reference count and resolver counts at the timing points indicated in fig 4.1 are given in the table, fig 4.2.

Timing Point	Ref. Encoder Count	Resolver Count	BIAS Command Line Which Determines Distance
T0	0	0	0 act.posit.1 = 0 1 act.posit.2 = 0
T1	16384	8192	10 position = 16384
T2	49152	40960	13 position = 32768
T3	65536	49152	17 position = 16384
T4	100000	0	29 [variable 0]=[variable 0] + 100000

Fig 4.2 – Relative Distances Between Line Encoder and Resolver Count

4.1 Completing Example Program

To achieve the home function the BIAS command 'move datum Mode X' can be utilised prior to the above code. Implementing the 'run' functionality simply requires monitoring of input 7.

Scaling can be used such that 'real-world' units can be implemented. These units are defined when completing the BIAS definition for the X40 input

A complete list of one potential solution appears in Appendix B.

Appendix A : Synchron Commands

A.1 Synchronadjust 1, Mode=n , Offset = [Variable X], StartOffset [Variable X]

Mode = n n=	Function
0	Normal synchron move
1	Ramp down in half Masterstroke(1) counts with a Startoffset(2)
2	Ramp up in half Masterstroke(1) counts with a Startoffset(2)
3	Reserved – special function
Mode + 32	Offset phase shift value will be loaded immediately, not on format trigger
Mode + 64	Offset phase shift value will be loaded as offset on Masteraxis(3)
Mode + 128	Slaveaxis(4) sychronous speed start position in 'Startoffset' parameter

Offset	Phase shift value on Slaveaxis(4) or on the Masteraxis(3)(Mode+64). The phase shift is an absolute value
Startoffset	Phase shift between the master encoder and the ramp up / down function. If the Mode+128 is selected the resolver count is used.

(1) – 'Masterstroke' The line encoder distance set by 'position = X' when during ramping phases of cycle.

(2) – 'Startoffset' Phase shift between line encoder and resolver required on detecting format trigger (status 2<>0) and starting ramp up / down function.

(3) – 'Masteraxis' The master axis is deemed to be the axis generating the line encoder signal.

(4) – 'Slaveaxis' The slave axis is deemed to be the axis being controlled by the servo controller. The slave count is that of the resolver.

A.2 Synchronadjust 2, Linear = x , Mode = y, Val. = z

Linear = x x=	Function
0	Actual position 1 will be reset to zero after every format(5)
1	Actual position 1 will not be reset after every format(5)

(5) = 'Format' Format is a generic term referring to the synchronous move cycle including the ramp up, the synchronous move and the ramp down operations.

Mode = y, y=	Value = z, z=	Function
1	0 to 500	Offset speed adjustment for phase shift on slave axis as a percentage of the actual synchronous speed
2	0 to ...	Offset speed adjustment for phase shift on slave axis in increments per sampling cycle
3	0	Bi-directional operation possible
	1	Bi-directional operation not possible
4	0	Rampfilter will effect the total position setpoint
	1	Rampfilter will affect on the offsetadjustment on slave axis in synchronous mode.
5	0 to 500	Offset speed adjustment for phase shift on master axis as a percentage of the actual synchronous speed
6	0 to ...	Offset speed adjustment for phase shift on master axis in increments per sampling cycle
7	0	Positive synchronous start direction
	1	Negative synchronous start direction

A.3 If Status X = Y then jump ...

X=	Y=	Function
0	0	Position not reached
	1	Position reached
1	0	Drive active
	1	Drive not active
2	0	Trigger format not reached
	1	Trigger format reached, set for 2msec
3	0	Trigger coupling not reached
	1	Trigger coupling reached, set for 2msec
4	0	Synchronous offset adjust running
	1	Synchronous offset adjustment reached
5	0	Profile executed, new profile number can be used
	1	Profile start trigger, set to 2msec
6	0	Last registration OK (mode 1, 3, 4, 5, 6 and 7)
	1	Error in last registration cycle (mode 1, 3, 4, 5, 6 and 7)
10	0	Movement in progress
	1	Ready for movement
11	0	No braking ramp
	1	Braking ramp in progress

Appendix B : Sample Solution

The following BIAS program could act as a suitable solution to the application as described.

```
PROG_START:
0     acceleration    = 500     rpm/s
1     deceleration    = 10000   rpm/s
2     speed            = 100    rpm
3     start axis
4     move datum;      mode = 8
WAIT_START:
5     If   input   9    == 0 then jump     WAIT_START
6     act. posit. 1 = 0            INKR
7     act. posit. 2 = 0            INKR
8     [variable 0  ] = 0
9     [variable 1  ] = 0
10    acceleration    = 50000   rpm/s
11    deceleration    = 50000   rpm/s
12    gear factor  = 1
13    synch.adjustment;     linear= 1 ,mode = 0    , val.= 1
14    synchr.settings;mode =130  offset=[var.1  ]; startoffset=[var.1  ]
15    [variable 0  ] = 100000
START_NEXT:
16    position =  16384        INKR
17    start axis
18    move synchron
19    position =  32768        INKR
20    update parameter
WAIT1:
21    If   status 2    == 0 then jump     WAIT1
22    output  20  = 1
23    position =  16384        INKR
24    update parameter
25    synchr.settings;mode =1  ; offset=[var.1  ]; startoffset=[var.1  ]
WAIT2:
26    If   status 2    == 0 then jump     WAIT2
27    output  20  = 0
WAIT3:
28    If   status 2    == 0 then jump     WAIT3
29    wait time    100    ms
30    start axis
31    move position;         v = 3000  rpm,  s= 0            INKR
32    wait for position reached
WAIT_MASTR:
33    If   actual pos. 2 <  [variable 0  ] then jump     WAIT_MASTR
34    synchr.settings;mode =130  offset=[var.1  ]; startoffset=[var.0  ]
35    [variable 0  ] = [variable 0  ] + 100000
36    If   input   7    == 1 then jump     START_NEXT
37    stop axis;     mode = 1
WAIT2MOVE:
38    If   input   7    == 0 then jump     WAIT2MOVE
39    jump      PROG_START
```