

Date: 13 October 2009

Subject: Synchronized Motion

White Paper

1. Synchronizing Servo Motion Axes

There are many multi-axis motion controllers on the market capable of synchronizing motion. Synchronous motion can be anything from coordinated motion of two or more motion axes in position lock, to triggered motion from an external event, to following a master reference as with flying shears, winders, electronic camming, and conveyor synchronizing. In this article, we will discuss various types of synchronized motion and implementation. Finding a single motion controller that can implement all types of synchronizing easily and quickly is often more challenging than programming the motion.

1.1. Electronic Gearing

Electronic gearing is by far the simplest synchronized motion. Ideally, this is when a slave mechanism is count-for-count position locked to a master reference at some ratio. The mechanism can be a rotary motor, linear motor or other actuator. The reference is typically a real or virtual encoding device. The most common example is the electronic line shaft. The electronic line shaft eliminates the mechanical linkage from a main shaft to one or more slave motors. In a multi-axis electronic line shaft the same can be accomplished using four motors and a reference signal. Figure 1 shows the electronic equivalent of a single master and four slaves. The ability to change connected ratios to any or all slave motors at any time is a key feature to electronic gearing. A simple example is shown below.

To connect four slave axes at various ratios to a master axis(0):

```
master=0
CONNECT(1.00, master) AXIS(1)
CONNECT(1.50, master) AXIS(2)
CONNECT(0.50, master) AXIS(3)
CONNECT(-1.25, master) AXIS(4) 'reverse motion
```

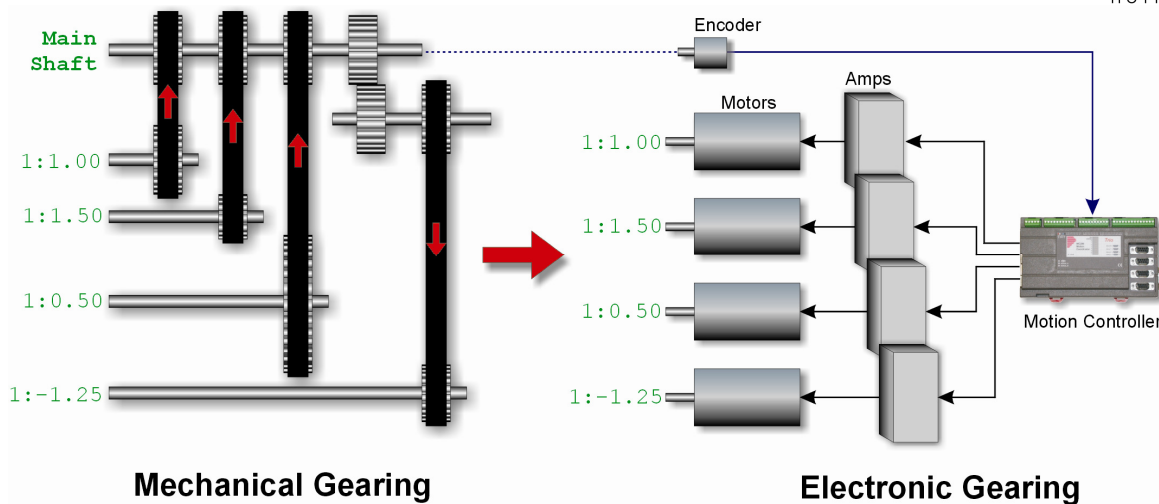


Figure 1

In some cases it may be desired to “clutch” a slave motor into gear lock (figure 2). This feature provides less mechanical shock when engaging and disengaging to a running line. To gear axis(2) to the master axis over 1 second:

```

master=0
BASE (2) 'Select the base axis
CLUTCH_RATE=1
CONNECT (1, master)

```

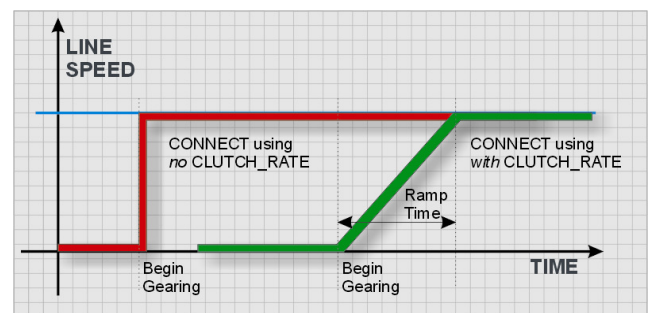


Figure 2

Another desired feature and big advantage of electronic gearing is the ability to phase shift (advance/retard) the position of any axis while maintaining ratio lock (figure 3). Most line shaft applications require some type of phase shifting of the motors by an operator. This can be done by use of a “virtual” or software axis and a few inputs. By simply performing the phase adjustment move profile on the virtual axis, it can be summed (or superimposed) to any slave axis. The result is a position-lock incremental move added or subtracted to the constant line speed in real time:

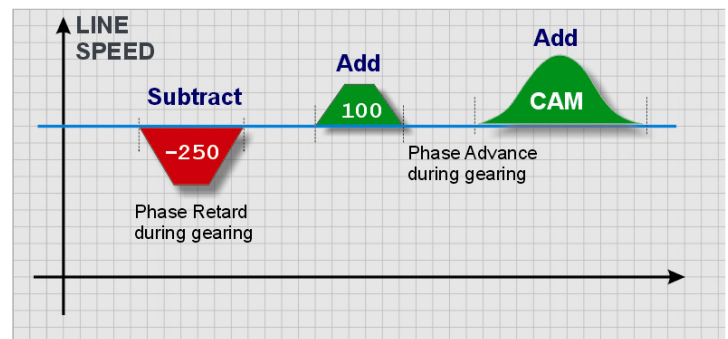


Figure 3

```

BASE (0) 'Master axis
ADDAX (7) 'Add 7 to master axis(0). 7 is the virtual axis
IF IN (1)=ON THEN MOVE (150) AXIS (7) 'Add 150 counts
IF IN (2)=ON THEN MOVE (-250) AXIS (7) 'Subtract 250 counts
IF IN (3)=ON THEN CAM (1,25,1,1000) AXIS (7) 'Make a CAM move

```

For a phase correction on axis(4) we can change the reference BASE(0) to BASE(4). The phase adjusted move made on the virtual axis(7) would then be applied to axis(4). Sometimes having an even gear ratio such as 1:1 or 1:2.5 is not possible or desirable.

Non-terminating ratios such as 1/3 (0.33333) will accumulate an error over time. A motion controller must be able to represent these ratios to solve the problem. If the ratio can be expressed as 2 integers, then it is possible to represent 0.33333. By using encoder counts as 1000/3000 where 1000 = slave movement and 3000 = master movement. The controller will calculate a new slave target each servo period that will always result in a 1/3 ratio between the two eliminating long term drift.

```
MOVELINK(1000,3000,0,0,master,4)
```

1.2. Linear and Circular Interpolation

Applications such as pick-n-place, assembly and cutters require the synchronization of multiple axes to get from A point to point B at the exact same time. Additionally, cutting and gluing applications require a constant vector or path speed. In an x-y-z mechanism, the motion controller must calculate this path speed as functions of x-y, x-z, and y-z motor speeds for true interpolation using Pythagoras:

$$\text{Vector_speed} = \sqrt{x_{\text{speed}}^2 + y_{\text{speed}}^2}$$

High-speed pick and place motion can be accomplished by simple looping. In figure 4, an example of repeat motion is shown for an x-y grid 1000mm x 1000mm. The center of each sub-block is set by 'xscale' and 'xmid'.

```
start:
  'Set up grid scaling
  xscale=1000/6
  xmid=xscale/2
  yscale=1000/6
  ymid=yscale/2

  FOR x=0 TO 5
    FOR y=0 TO 5
      GOSUB pick
      MOVEABS((x*xscale)+xmid,(y*yscale)+ymid)
      WAIT IDLE
      GOSUB place
    NEXT y
  NEXT x
  GOTO start
```

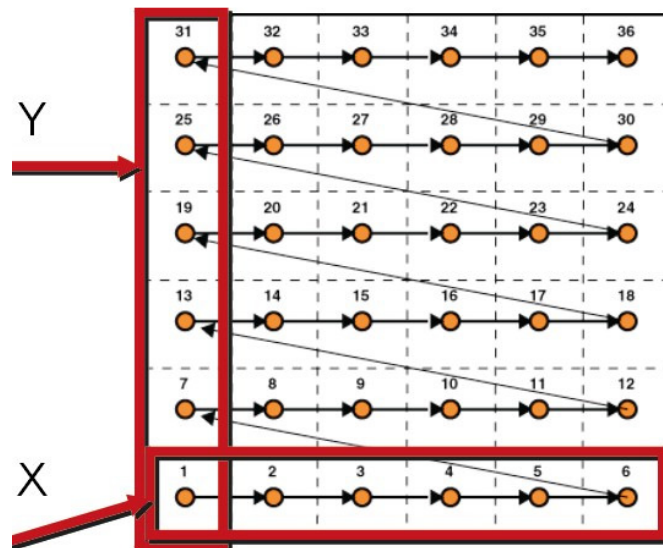


Figure 4

The pick and place motion is a series of absolute move steps designed for fast starts and stops. For contoured motion to cut or lay glue we need to define a path and path speed. The path can be a series of lines and arcs then seamed together as one move. Figure 5 shows a gasket operation using this technique.

```

BASE(0,1)      'Set the Axes of interpolation
SPEED=1200    'Set vector speed (set by axis(0))
MERGE=ON      'Merge all moves together
MOVE(0,30)
MOVECIRC(10,10,10,0,1) 'Relative arc from last move
MOVECIRC(10,10,0,10,0)
MOVECIRC(10,10,10,0,1)
MOVE(30,0)
:
:

```

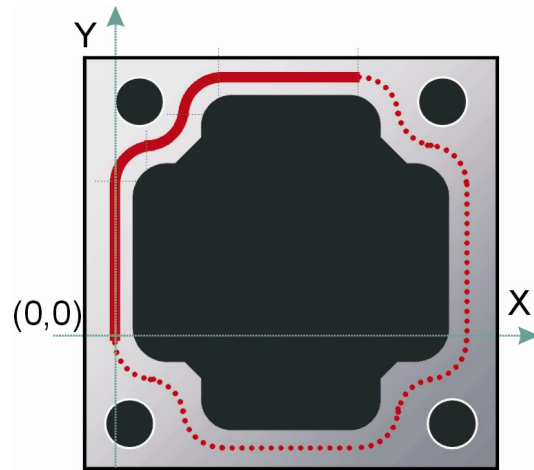


Figure 5

1.3. Coordinate Transformation

In addition to the standard one-to-one x-y multi-axis interpolation, coordinate transformations allow movement to be specified in a coordinate frame of reference that does not correspond one-to-one with the axes.

For example, a common 2D mechanism is a single-belt arrangement (figure 6). Traditionally, the x or y motors move to produce motion in either dimension. This is not the case in the single-belt x-y system and therefore requires a transform of coordinates:

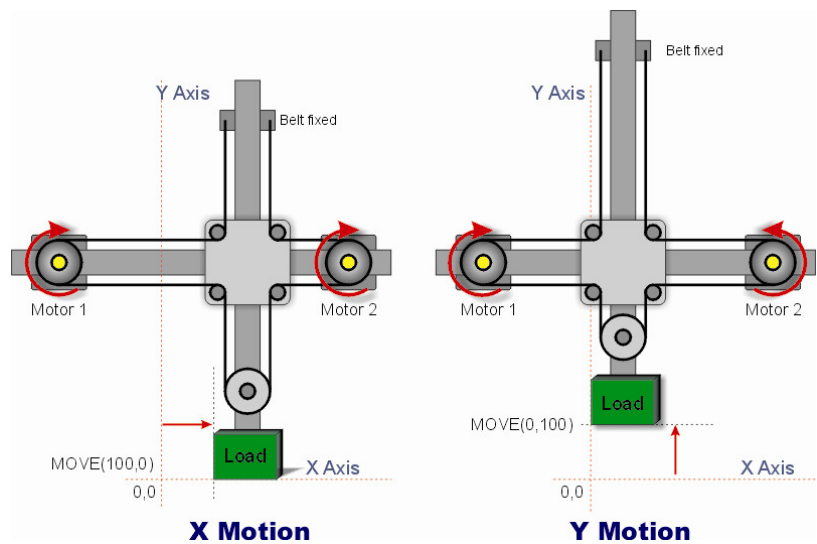


Figure 6

$$X_{transformed} = (X_{position} + Y_{position}) * 0.5$$

$$Y_{transformed} = (X_{position} - Y_{position}) * 0.5$$

Switching to the new “frame” will allow x-y motion in the new coordinate system using the same MOVE commands (see illustration). In this mode, an interpolated move of MOVE(0,100) or MOVE(100,0) produces motion on both motors to move the load vertically or horizontally, based on the transformed position. By defining the coordinate equations for any special mechanism such as a 3-axis scara robot, a new FRAME can be selected.

```

BASE(0,1)      'Set the Axes of interpolation
FRAME=2       'Set new coordinate system
MOVE(0,100)   'Make coordinated move in new frame

```

1.4. Synchronizing Events to Trigger Motion

Another common way to synchronize motion is triggering from an external event or position. Typical applications for triggered motion include labels from a registration mark, press-feeds, and flying shears. In practice, it can be any application that requires very accurate capturing or resetting of encoder position on-the-fly for some operation thereafter whether it’s cutting, gluing, or placing.

The way a controller handles the high-speed position capture to generate motion, however, can become critical as speeds increase. Some controllers can latch the axis encoder position in hardware (1µs or less) for minimum delay. While the captured position is accurate to the external event, typical command generation and execution to respond to the event can take several orders of magnitude longer (up to 1ms) or the next servo period. This is simply not fast enough for most high-speed machines. If the start of the motion profile could be “recalibrated” to the external event the controller could make up for time lost since the event.

To illustrate the significance let’s look at a hypothetical high-speed label feeder. A conveyor is used to transport boxes onto which labels must be applied (figure 7). The boxes may be irregularly spaced and the labels must be synchronized and applied via a roll-feed mechanism at the line speed of the conveyor. A proximity sensor on the conveyor is connected to the controller’s registration input to sense the leading edge of the boxes. An optical sensor on the label feeder is connected to another registration input to allow us to compensate for any slippage or positioning errors on the label feed. The labels need to be applied at a known position relative to the leading edge of the box defined by a user specified offset called “**label_pos**”.

On the conveyor, the boxes arrive at random times. The proximity sensor can be used to recognize the presence of a box and to reset the position counter at the leading edge to zero. The ability to capture the position of the box edge, then offsetting to zero is key to placing the labels accurately especially at high speeds. Therefore, after the registration event has occurred, the measured position will actually reflect the absolute distance from the start of the box.

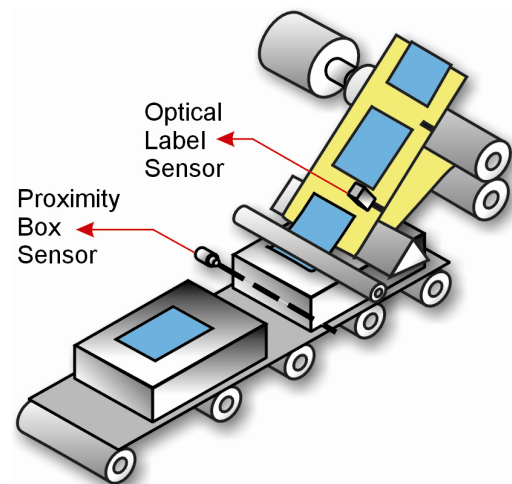


Figure 7

```

BASE(conveyor)
REGIST(3)      'Set to look for REG input (Box edge)
WAIT UNTIL MARK
OFFPOS = -REG_POS 'On-the-fly position adjust

```

On the label feeder, the registration mark printed on the label web is used to check the feed and correct for any errors. In this type of application, the registration control works by measuring any error on the current move, and applying a correction to the next. The controller must have ability to capture the label registration event which is the label position, and synchronize the label feed to the conveyor encoder for accurately positioning on the box. As the line speed increases command execution lag time can become an issue.

By capturing the label position in hardware using an external label sensor the lag time can be compensated. The position captured is accurate to the real external event but unfortunately motion command generation occurs slower at the servo period rate – a big problem at high speeds. To overcome this technical hurdle some controllers calculate the error between the captured position and actual motor position, and then add it to the motion command in the next servo period. So at $2t$ in figure 8, $132-127=5$ position units will get added to the next move update. This compensation technique ensures the label move will be as accurate as possible, regardless of conveyor speed changes.

```

BASE(label_feed)
Loop:
  Label_pos=127
  REGIST(3) 'Set to look for REG input (Label Reg mark)
  MOVELINK(lab_pitch+radj,linkdst,20,20,conveyor,2,label_pos)
  WAIT UNTIL MARK OR MTYPE=0 'Wait for MARK or MOVELINK is finished
    ' Reg mark seen, calculate error
  reg_err=expected-REG_POS
  radj=reg_err*0.4 'Adjust by reg.'gain'
  .
  .
GOTO loop

```

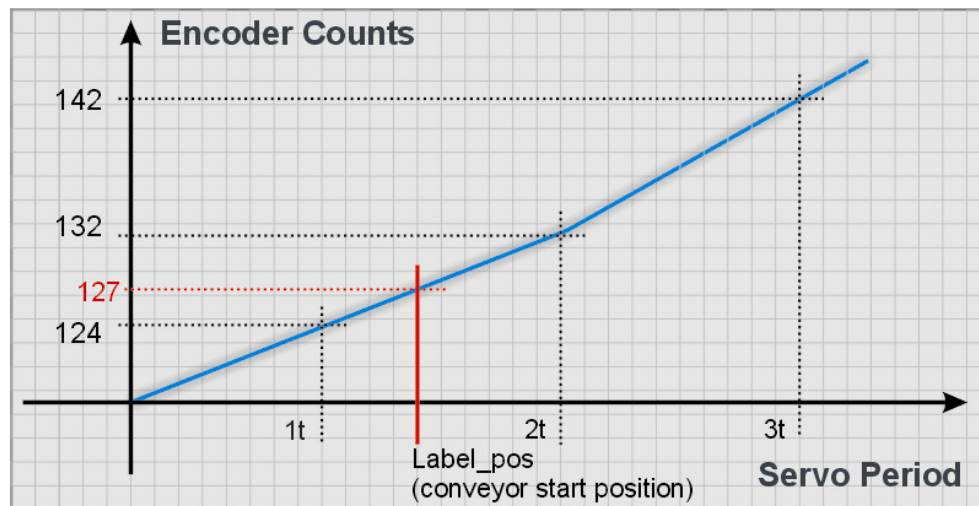


Figure 8

In summary, there are many different ways to synchronize motion from electronic gearing with phase shifting, to multi-axis interpolated motion, to registration capture and command generation. Understanding how a motion controller handles all types of synchronizing can save you time and possible applications problems in the long run when throughput speeds increase.