

PositionServo Sample Program - Simple Directional Control

Concept:

This sample program uses much of the features and functionality of the Standard Internal Velocity Mode program (AE0003). Please familiarize yourself with the Standard Internal Velocity Mode example before starting this sample program.

This sample program uses the PositionServo in 'Internal Velocity Mode'. The program uses two manual push buttons, B1 and B2, to cause motion in either the clockwise or counter clockwise direction. The velocity used for each manual move (commanded by the two push buttons) is set with the Analog input on Ain1. The program also utilizes the predefined hardware limit switch functionality within the drive by activating it through the parameter variables. A graphical representation of the axis is shown in Figure 1.

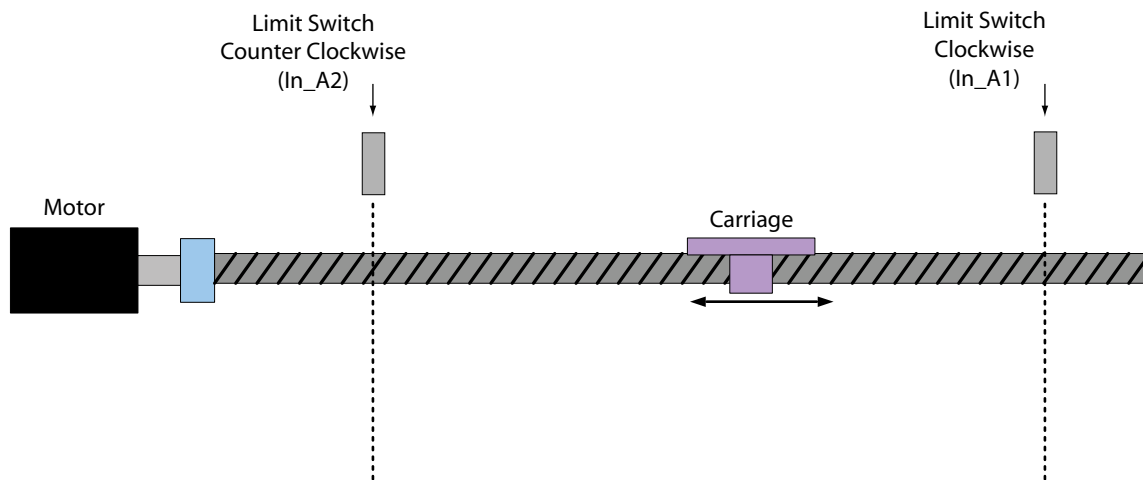


Figure 1: Limit Switch Location

The program uses the Enable Input on A3 as a safety enable for safety devices on the system. Input A4 is defined as the Run / Stop input to cause the drive to start / stop.

Input B1 causes motion in the counter clockwise direction, and input B2 causes motion in the clockwise direction. If neither button is pressed, or if both buttons are pressed simultaneously then the output velocity to the motor should be 0.

Digital output 1 provides an inverse fault indication (0 = Fault, 1 – Healthy) to the outside world.

Analog Input 1 is used as the velocity reference and is accessed through drive variable AIN1 (in volts: +10 to -10). Note that Analog input 1 scaling, dead-band, and offset (as set in MotionView or through associated variables) are not applied to the AIN1 variable and this functionality has to be created within the user program.

The PositionServo is placed in Velocity mode with internal reference through the relevant variables at the start of the program.

The program uses IREF (internal Reference) as its velocity reference when in internal velocity mode. Therefore Analog input 1 (AIN1) is transferred to IREF within the main program loop. AIN1 is in volts and IREF in RPS (revolutions per second) so AIN1 must be scaled before it can be applied to IREF. This program uses the analog scaling entered in the MotionView Parameter as this allows the customer to make parametric changes to the scaling from within MotionView without having to alter the user program.

Lastly the program applies a dead-band in RPS before the calculated velocity is transferred to the Internal Reference variable and is output to the motor.

Logic determines which push buttons are currently active to determine the direction of travel or to reset the velocity calculation to 0.

The drive will continually execute this code loop until the drive run input is switched off.

An event is used to detect the run / stop input transitioning from run to stop once the drive is in a run condition.

No subroutines are used in this example.

Motor Mechanics:

For the purpose of the demonstration a motor should be used with a disc fitted to the motor shaft so that the user can see the Velocity output executed by the PositionServo.

Fault Handling:

In the event of a fault, the code will be restarted. The operator must switch the drive run /stop input off and on again for the Velocity loop to restart. The drive healthy output is processed within the fault handler and subsequent recovery code.

I/O:

- IN_A1: Hardware Limit Switch – Clockwise
- IN_A2: Hardware Limit Switch – Counter Clockwise
- IN_A3: Safety Enable / stop - Connected to machine safety Guards / Devices
- IN_A4: System Run / Stop Input
- IN_B1: Run Counter Clockwise Input – Manual operator push button
- IN_B2: Run Clockwise Input – Manual operator push button
- OUT1: Drive Tripped – Fault / Healthy Output
- AIN1: Drive Velocity Reference

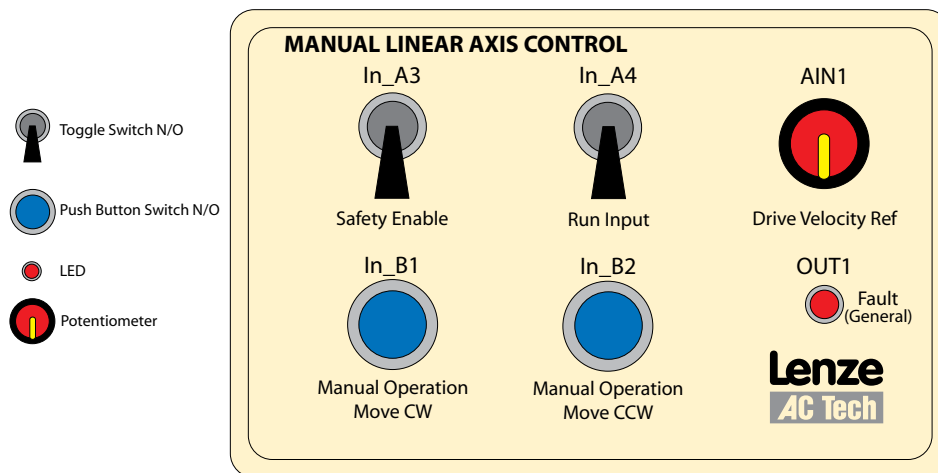


Figure 2: Manual Linear Axis Control

Connection:

Figure 3 illustrates the P3 terminals that need to be connected for this example to work.

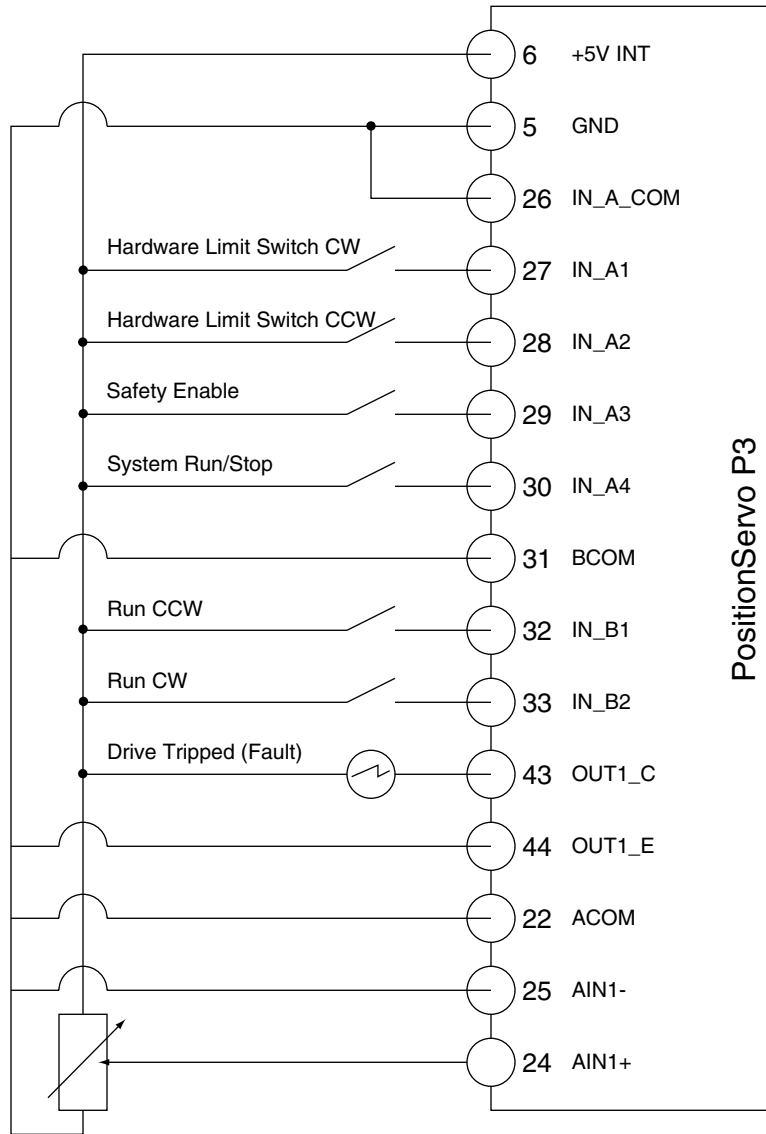


Figure 3: Connection Diagram- Simple Directional Control

Example Program:

The program code is color-coded for quick recognition of the various parts of the indexer program. The color coding is not in accordance with, or representative of, any national or international standard.

```

;***** PositionServo User Indexing Program *****
;***** Header *****
;Title      :      Simple Push Button Manual Direction Control
;Author     :      AC Technology International Ltd
;Description :      Program runs from analog voltage reference (AIN1) transferred to IREF Variable
;           :      Direction is determined by input B1 on for CW - B2 on for CCW
;           :      RUN Input provided on A4
;Version Number :      V1.0.1
;Date      :      25/11/06
;
;***** I/O List *****
;  Input A1 -      Hardware Limit Switch clockwise
;  Input A2 -      Hardware Limit Switch counter-clockwise
;  Input A3 -      Safety stop
;  Input A4 -      Run/stop
;  Input B1 -      Run Clockwise
;  Input B2 -      Run Counter Clockwise
;  Input B3 -      not used
;  Input B4 -      not used
;  Input C1 -      not used
;  Input C2 -      not used
;  Input C3 -      not used
;  Input C4 -      not used
;
;  Output 1 -      Drive Tripped Output (Event)
;  Output 2 -      not used
;  Output 3 -      not used
;  Output 4 -      not used
;
;  Analog In 1 -   Analog Speed Reference
;  Analog In 2 -   not used
;  Analog Out  -   not used
;
;  Encoder Out -   not used
;
;***** Initialize and Set Variables *****
; Define Constants and Variables. Assign I/O and Initialize Variable Values

  UNITS = 1                ; Units in RPS
Define Vel_Calc V0        ; Define Variable for Velocity Calculations
Define But_Run_CW IN_B1   ; Define name for clockwise input B1
Define But_Run_CCW IN_B2  ; Define name for clockwise input B2
Define Run_Signal IN_A4   ; Define name for run signal on input A4
Define Drive_Status OUT1  ; Define name for drive status digital output

VAR_HLS_Mode = 1         ;Defines Functionality of Hardware Limits as 'Fault' for Input Switches A1 and A2
VAR_REFERENCE = 1        ;set Reference to Internal
VAR_DRIVEMODE = 1        ;Set Operating mode to Velocity mode
VAR_ENABLE_SWITCH_TYPE = 1 ;enable switch function set to "Inhibit"

;***** Events *****
;Event to detect RUN input (A4) going off
Event Run_Input Run_Signal == 0
  Disable
  Jump Program_Start
Endevent

```

```

;***** Main Program *****

Out1 = 1 ; Fault status 1 = ok, 0 = Fault
Var_Enable_acceldecel = 1 ; Enable Ramps
Var_Accel_limit = 300000 ; Set Accel Ramp Rate
Var_decel_limit = 300000 ; Set Decel Ramp Rate

PROGRAM_START:
Event Run_Input off

IREF = 0

Wait While Run_Signal == 0 ; Wait for run input to come on
Enable ; Enable Drive
Event Run_Input On ; Turn on event to detect run input going off

VELOCITY_LOOP:
; Update the velocity reference
Vel_Calc = (ain1 * var_velocity_scale / 60)

if But_Run_CW == 1 && But_Run_CCW == 0
    Vel_Calc = Vel_Calc ; Move Clockwise command - don't change the calculation
endif

if But_Run_CW == 0 && But_Run_CCW == 1
    Vel_Calc = Vel_Calc * -1 ; Move Counter-Clockwise command - change direction of the calculation
endif

if But_Run_CW == 0 && But_Run_CCW == 0
    Vel_Calc = 0 ; Invalid Input - don't allow movement
endif

if But_Run_CW == 1 && But_Run_CCW == 1
    Vel_Calc = 0 ; Invalid Input - don't allow movement
endif

If Vel_Calc <= 0.6 && Vel_Calc >= -0.6 ; Add If Statement to Include Deadband
    Vel_Calc = 0;
Endif

IREF = Vel_Calc

GOTO VELOCITY_LOOP
END

Fault_Reset:
    wait until Run_Signal == 0 ; Wait until Drive run input disabled
    Drive_Status = 1 ; Fault status 1 = ok, 0 = Fault
Goto Program_Start

;***** Sub-Routines *****
; Enter Sub-Routine code here

;***** Fault Handler Routine *****
; Enter Fault Handler code here
ON FAULT

    Drive_Status = 0 ; Fault status 1 = ok, 0 = Fault
    Resume Fault_Reset

ENDFault

```