```
PROGRAM  PID_ST
VAR
    // Integral gain for position loop PID
    lILocal :  LTIME ;

    // Definition of PID contoller type 03 Velocity loop
    PID_Velocity :  IL_PIDType03 ;
    // Definition of PID contoller type 03 Position loop
    PID_Position :  IL_PIDType03 ;

    // Definition of IIR filtertype 01
    LP_Filter_Position_Sensor :  IL_IIRType01 ;
    fbIIR1 :  CXA_LOOPCONTROL . IL_IIRType01 ;  // Declaration of filter type
    tIIR_Coff :  CXA_LOOPCONTROL . IL_IIR_COEFF ; // Configuration of filter
coefficient


    // Output from position loop PID to velocity setpoint
    rPIDOut :  LREAL ;

    rdt :  LREAL  :=  0.002 ;  // time step for derivation
    rVelocity  :  LREAL ;  // Velocity output from derivation of ball position
feedback signal
    position  :  LREAL ;  // stored position, used for derivation
    prev_pos  :  LREAL  :=  0.0 ;  // previous position, used for derivation
    // Output from filtering of the ball position feedback
    rLP_PosFilter_out :  LREAL ;
    // Angle generator for the sinus function
    AngleGenerator :  IL_AngleGeneratorType01 ;
    // Sinus signal generator
    SinusSignal :  IL_SignalGeneratorType01 ;
    // Output from angle generator to sinus generator angle input
    rSinusAngle :  LREAL ;
    // Output from sinus generator to amplitude calculation
    rSinusOut :  LREAL ;
    // kp gain for Position loop PID
    rPLocal :  LREAL ;
    // Derivative gain for position loop PID
    lDLocal :  LTIME ;
    // kp gain for Velocity loop PID
    rPLocalVel :  LREAL ;
    // Integral gain for velocity loop PID
    lILocalVel :  LTIME ;
    // Derivative gain for velocity loop PID
    lDLocalVel :  LTIME ;
END_VAR

```

```
// Ball position control program


// Assigning global variables
GVL . rActualBeamPos  :=  arAxisStatus_gb [ 1 ] . Data . ActualPosition ;
gvl . rFollowingError  :=  ABS ( gvl . rBallPosCmd  -  gvl . rBallFeedback ) ;

// Setting local position loop pid variables to global variables from HMI
// rPLocal := gvl.rPGainPID; // reset DControl to original value
rPLocal  :=  gvl . rPGainPID / 1000 ;  // reset pControl to original value
lILocal  :=  gvl . lICtrlPID ;  // reset iControl to original value
lDLocal  :=  gvl . lDCtrlPID ;  // reset DControl to original value
// Setting local velocity loop pid variables to global variables from HMI
rPLocalVel  :=  gvl . rPGainPIDVel ;  // reset pControl to original value
lILocalVel  :=  gvl . lICtrlPIDVel ;  // reset iControl to original value
lDLocalVel  :=  gvl . lDCtrlPIDVel ;  // reset DControl to original value

// IIR filter coefficients from c2d conversion in matlab. Forward coefficients
are the numerator values, b0,b1,b2.., and
// the Backward coefficients are the denominator values, 1,a1,a2....
// Single LP filter at frequency= 8.8 rad/s
tIIR_Coff . BackwardCoeffs [ 1 ] :=  gvl . rBwdCoeff1 ;
tIIR_Coff . BackwardCoeffs [ 2 ] :=  gvl . rBwdCoeff2 ;
tIIR_Coff . ForwardCoeffs [ 0 ] :=  gvl . rFwdCoeff1 ;
tIIR_Coff . ForwardCoeffs [ 1 ] :=  gvl . rFwdCoeff2 ;
tIIR_Coff . NumberOfCoeffs  := 4 ;

//------------------------------------------------------------------------------
// Generator for sinus signal for the sinus function in the HMI
AngleGenerator (
        Enable :=  gvl . bEnablePID ,
        InOperation =>  ,
        Error =>  ,
        ErrorID =>  ,
        ErrorIdent =>  ,
        Pause :=  ,
        Frequency :=  gvl . rSinusFreq ,
        InPause =>  ,
        OutputAngle =>  rSinusAngle ,
        ActScanTime =>  ) ;


    SinusSignal (
        Enable :=  gvl . bEnablePID ,
        InOperation =>  ,
        Error =>  ,
        ErrorID =>  ,
        ErrorIdent =>  ,
        Angle :=  rSinusAngle ,
        CurveType :=  1 ,
        Duty :=  gvl . rSinusDuty ,
        DutyRamp :=  25 ,
        Rounding :=  ,
        RoundingRamp := ,
        OutputValue =>  rSinusOut  ,
        ActScanTime =>  ) ;

        // Sinus output to ball position control PID;
        gvl . rSinusOutputPID  :=  ( rSinusOut ) * gvl . rAmplitude + 330 ;

//-------------------------------------------------------------------------------
// Low pass filter for filtering ball position feedback signal
LP_Filter_Position_Sensor (
    Enable :=  gvl . bEnablePID ,
    InOperation =>  ,
    Error =>  ,
    ErrorID =>  ,
    ErrorIdent =>  ,
    Pause :=  ,
    Value :=  gvl . rBallFeedback ,
    StartValue :=  ,
    Coefficients :=  tIIR_Coff ,
    InPause =>  ,
    ActScanTime =>  ,
    OutputValue =>  rLP_PosFilter_out ) ;
//-----------------------------------------------------------------
// PID function block for position closed loop
PID_Position (
    Enable :=  gvl . bEnablePID  ,
    InOperation =>  gvl . bPIDActive  ,
    Error =>  gvl . bPIDError ,
    ErrorID =>  ,
    ErrorIdent =>  ,
    Pause :=   ,
    Preset :=  ,
    Setpoint :=  gvl . rBallPosCmd  ,
    Feedback :=  rLP_PosFilter_out ,
    PresetValue :=  GVL . rPresetVal ,
    HighLimit :=  GVL . rCtrlMax ,
    LowLimit :=  GVL . rCtrlMin ,
    PControl :=  rPLocal ,
    IControl :=  lILocal ,
    DControl :=  lDLocal ,
    bControl :=  gvl . rbCtrlPID ,
    cControl :=  gvl . rcCtrlPID ,
    InPause =>  ,
    PresetAck =>  ,
    ActScanTime =>  ,
    HighLimitActive =>  ,
    LowLimitActive =>  ,
    ControlValue =>  gvl . rPIDOut  ) ;
//--------------------------------------------------------------------
// Finding the derivative from the position signal to get velocity
// Calculate velocity using derivation, rdt is the cycle time 0.002s

```

```
gvl . rVelocity  :=  ( ( rLP_PosFilter_out / 1000 )  -  prev_pos )  /  rdt ;

// update previous position
prev_pos  :=  ( rLP_PosFilter_out / 1000 ) ;
//----------------------------------------------------------------------
// PID fucnction block for velocity closed loop
PID_Velocity (
    Enable :=  gvl . bEnablePID  ,
    InOperation =>  gvl . bPIDActive  ,
    Error =>  ,
    ErrorID =>  ,
    ErrorIdent =>  ,
    Pause :=   ,
    Preset :=  ,
    Setpoint :=  gvl . rPIDOUT ,  // rPIDOut_Filter
    Feedback :=  gvl . rVelocity ,
    PresetValue :=  GVL . rPresetVal ,
    HighLimit :=  GVL . rCtrlMax ,
    LowLimit :=  GVL . rCtrlMin ,
    PControl :=  rPLocalVel ,
    IControl :=  lILocalVel ,
    DControl :=  lDLocalVel ,
    bControl :=  gvl . rbCtrlPID ,
    cControl :=  gvl . rcCtrlPID ,
    InPause =>  ,
    PresetAck =>  ,
    ActScanTime =>  ,
    HighLimitActive =>  ,
    LowLimitActive =>  ,
    ControlValue =>   gvl . rPIDCtrlOut ) ;
```