Pyramid Technical Consultants, Inc.
1050 Waltham Street, Suite 200, Lexington, MA 02421
TEL: (781) 402-1700 ♦ FAX: (781) 402-1750 ♦ URL: WWW.PTCUSA.COM

Document Name			
PTC_ControlsLV LabView VI Library			
Written by	Version	Create Date	Modified on
Christopher J Pendleton cpendleton@ptcusa.com	1.0	03/19/2008	

TABLE OF CONTENTS

1 Introduction

1.1 Overview

PTC_ControlsLV is a LabView interface to the standard PTC software interface DLL, PTC_Controls32. PTC_Controls32 is a win32 C++ class library that provides object-oriented control and access to PTC hardware objects. Because LabView does not provide comprehensive support for C/C++ style pointers, it was necessary to use a layer between these two modules. Since LabView now provides a quality Microsoft .net interface, this layer was developed in .Net. The .Net software module is called Interpreter.dll, and must reside in the same directory as PTC_ControlsLV.llb and PTC_Controls32.dll. Another .Net module, PTCScreenControls.dll, is a dependency of Interpreter, and must also be in this directory.

PTC_ControlsLV is presented a VI library. Each VI represents a function call that exists in PTC_Controls32. The VI library is designed as a pseudo object oriented system. Certain classes of PTC hardware objects inherit functionality from parent classes. The PTC_ControlsLV class hierarchy is show in the following class digrams.

2 Architecture

2.1 Class Structure

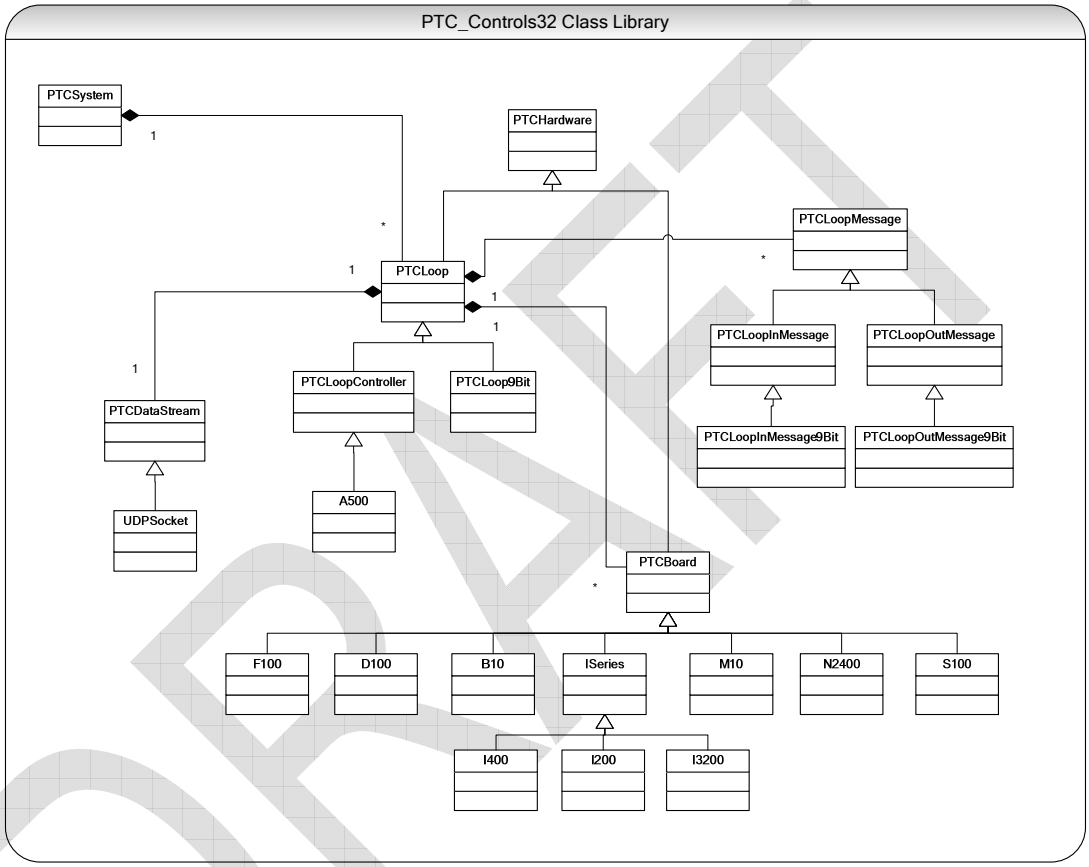

Figure 1 - Class Diagram

3 PTC_ControlsVI VI Library

The following classes and functions will be available to the user to control PTC devices.

3.1 Class PTCSystem

3.1.1 PTCSystem_LoadSystem(string xmlFile)

Arguments:

1. *xmlFile*	XML representation of the A500 system of looped devices. An example of this XML configuration is shown in the appendix.

Returns:

1. *success*	True/false value indicating whether the system file was successfully loaded.
2. *PTCSystemLink*	Reference to the created PTCSystemLink, if the function was successful, null otherwise.

Description:

This function reads in an XML system configuration file that creates the A500 and device objects. It assigns a name to every object. PTCSystem can later hand these objects to the user application when a name is specified. See appendix for more explanation and example.

3.1.2 PTCSystem_StartSystem(PTCSystemLink system)

Arguments:

1. *system*	PTCSystemLink reference.

Returns:

1. *succes*	True/false value indicating whether the system file was successfully started.

Description:

Initializes and starts each A500 and device.

3.1.3 bool PTCSystem_StopSystem(PTCSystemLink system)

Arguments:

1. *system* PTCSystemLink object reference.

Returns:

1. *success* True/false value indicating whether the system file was successfully stopped.

Description:

Shuts down each A500 and device. Call this on system shutdown.

3.1.4 PTCSystem_GetHardwareLink(PTCSystemLinksystem, string name)

Arguments:

1. *system* PTCSystemLink object reference.
1. *name* Name of the desired PTCHardwareLink device, as specified in the PTCSystem XML configuration file.

Return:

1. *PTCHardwareLink* Reference to the PTCHardwareLink, if the function was successful, null otherwise.

Description:

Gets a specified PTCHardwareLink object reference by name. This object must have been described in the PTCSystem XML configuration file.

3.2 Class PTCBoard (abstract)

3.2.1 PTCBoard_Initiate(PTCHardwareLink board, uint32 cycleNumber)

Arguments:

1. *board* PTCHardwareLink object reference.
2. *cycleNumber* Sets the cycle number of the acquisition.

Returns:

1. *error* True/false value indicating whether the specified PTCBoard had an error.
2. *errorCode* Error code, if an error exists.

Description:

This function arms the device for measurement. Measurement will start when the device is triggered. Measurement will stop after the set number of trigger points is taken or a hardware signal stops the acquisition.

3.2.2 PTCBoard_Abort(PTCHardwareLink board)

Arguments:

1. *board* PTCHardwareLink object handle.

Returns:

1. *error* True/false value indicating whether the specified PTCBoard had an error.
2. *errorCode* Error code, if an error exists.

Description:

This function disarms the device for measurement.

3.2.3 PTCBoard_Connect(PTCHardwareLink board)

Arguments:

1. *board* PTCHardwareLink object handle.

Return:

1. *error* True/false value indicating whether the specified PTCBoard had an error.
2. *errorCode* Error code, if an error exists.
3. *Device Version* Version of the device PIC firmware.
4. *FPGA Version* Version of the FPGA firmware.
5. *Device Type* Device type (from enumeration).
6. *Hardware Version* Device hardware rev

Description:

This function puts the device in the connected state and returns the device attributes to the user application.

3.2.4 PTCBoard_GetDataAvailable(PTCHardwareLink board)

Arguments:

1. *board* PTCHardwareLink object handle.

Return:

1. *data available* True/false value indicating whether the specified PTCBoard has data available to be read

Description:

This function returns whether the device has new data available to be read.

3.3 Class ISeries (abstract, inherits from PTCBoard)

3.3.1 ISeries_SetConfiguration(PTCHardwareLink iseries, CLUSTER [uint8 capacitor, uint8 mode, uint8 source, uint32 points, uint8 polarity, float time])

Arguments:

1. *iseries* PTCHardwareLink object reference
2. *capacitor* Desired integration capacitor. Valid values 0 or 1. Values of the capacitors can be found in the PTC_ValueISeries structure that is called back with the GetValue command.
3. *mode* Desired accumulation mode of the ISeries device. Valid values are 0-3.
   - Value 0 = None
   - Value 1 = No Compensation
   - Value 2 = Estimated
   - Value 3 = Lossless
4. *source* Sets the trigger source. Valid values are found in the PTC_TriggerSource enumeration.
5. *points* Sets the number of readings to take after the device is triggered. Set to maximum uint32 for infinite readings.
6. *polarity* Desired polarity of the signal edge to trigger on.
   - Value 0 = rising
   - Value 1 = falling
7. *time* Sets the integration period of the device. Limits?

Returns:

1. *error* — True/false value indicating whether the specified PTCBoard had an error.
2. *errorCode* — Error code, if an error exists.

Description:

This function sets multiple parameters of an ISeries device.

Relevant Enumerations:

```
enum PTC_TriggerSource
{
    Internal = 0x01,
    ExternalStart = 0x03,
    ExternalStartStop = 0x05
}
```

3.4 Class I3200 (inherits from ISeries)

3.4.1 I3200_GetData(PTCHardwareLink i3200)

Arguments:

1. *i3200* — PTCHardwareLink object reference

Returns:

1. *error* — True/false value indicating whether the specified PTCBoard had an error.
2. *errorCode* — Error code, if an error exists.
3. *CLUSTER (if no error)*
   a. *cycleNumber* — Cycle number, as passed into PTCBoard_Initiate.
   b. *data* — 32 channel array of I3200 charge data.
   c. *integrationNumber* — Integration number since trigger.
   d. *subsampleNumber* — Subsample number within current integration.
   e. *timeSinceTriger* — Time, in us, since trigger.
   f. *timeSinceSettle* — Time since last integration settle.

g. *Overrange flags* — 32 bits of channel overage flags. 1=overranged, 0 = not overranged.

h. *sampleCount* — Sample count, since trigger.

Description:

This function gets the next I3200 data sample.

3.4.2 I3200_GetOutput(PTCHardwareLink i3200)

Arguments:

1. *i3200* — PTCHardwareLink object reference

Returns:

1. *error* — True/false value indicating whether the specified PTCBoard had an error.
2. *errorCode* — Error code, if an error exists.
3. *CLUSTER (if no error)*
   a. *capacitor* — Currently selected capacitor, 0 or 1
   b. *accumulationMode* — Current accumulation mode
   c. *calibrationSource* — Channel with internal calibration set, or 0.
   d. *integrationPeriod* — Current integration period, in amperes.
   e. *range* — Current full scale range, in A.
   f. *subsamples* — Current number of subsamples selected.
   g. *holdDuringMux* — Hold analog value when muxing
   h. *triggerConnector* — Trigger gate signal connector
   i. *triggerDelay* — Trigger delay
   j. *triggerPoints* — Trigger points selected
   l. *polarity* — Trigger polarity selected
   m. *source* — Trigger source selected
   n. *externalBias* — Current external bias, if available
   o. *relayActuated* — 1=actuator on
   p. *continuity* — 1=on, 0=off

q. *resetTime*	Current reset times, per capacitor
r. *settleTime*	Current settle times, per capacitor
s. *setupTime*	Current setup time
t. *progGain*	PGA gain setting

Description:

This function gets the current I3200 output values.

3.5 Global Types

3.5.1 Enumerations

```
// Error codes
enum PTC_Error
{
        Processing = 12,
        StatusError = 11,
        A500LoopBreakError = 10,
        MotionControllerError = 9,
        MotionControllerCommsError = 8,
        InvalidDevice = 7,
        InvalidVersion = 6,
        FileError = 5,
        Timeout = 4,
        BadEcho = 3,
        BadTag = 2,
        BadChecksum = 1,
        None    = 0,
        UndefinedHeader = -113,
        UnexpectedNumberOfParameters = -115,
        NumericDataError = -120,
        InvalidcharacterInNumber = -121,
        NotConnected = -200,
        CommandProtected = -203,
```

```
        SettingConflict = -221,
        DataOutOfRange = -222,
        IllegalParameterValue    = -224,
        DataCorruptOrStale = -230,
        HardwareError = -240,
        TxRamFailure = -242,
        RxRamFailure = -243,
        FmRamFailure = -244,
        HardwareMissing = -241,
        CorruptMedia    = -253,
        OutOfMemory    = -291,
        DeviceBusy = -311,
        ProgramNotLoaded = -335,
        QueueOverflow = -350,
        DataNotAvailable = -401,
        Overflow = -901,
        _Timeout = -902,
        Address = -903,
        MsgLength = -904,
        DeviceTypeConflict = -905,
        Checksum = -906,
        ProgramNotFound = -907,
        FlashError = -908,
        StateConflict = -909,
        InvalidLoop = -910,
        LocalMode = -911,
        _InvalidDevice = -912,
        NotYetSupported = -913,
        DeviceNotReady = -914
};
```

4 Appendix

4.1 PTCSystem XML Configuration File

The A500 system and all devices will be configured with an XML file representing the system and all of its relevant attributes. This is done with the PTCSystem::LoadSystem command. The table of xml elements, associated attributes, and allowed quantity is below.

element	level	attributes	# allowed
system	0	none	1
loopcontroller	1	1. type (always use "A500") 2. name (string) 3. ip (valid IP address) 4. port (0 - 65535)	5
loop	2	1. name (string) 2. number (1-5, unique)	5
board	3	1. type ("I3200"/"I200"/"M10"/N2400"/"S100") 2. name (string) 3. address (1-15)	15

An example file is shown below.

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<system>
        <loopcontrollers>
                <loopcontroller type="A500" name="MWPC_A500" ip="192.168.100.4" port="10000">
                        <loops>
                                <loop number="1" name="MWPC_A500_LOOP1">
                                        <boards>
                                        <board type="I3200" name="MWPC_I3200_1" address="6" />
                                        <board type="S100" name="MWPC_S100_1" address="7" />
                                        </boards>
```

```
                                    </loop>
        <loop number="3" name="MWPC_A500_LOOP3">
          <boards>
            <board type="S100" name="MWPC_S100_2" address="4" />
            <board type="N2400" name="MWPC_N2400_1" address="6" />
          </boards>
        </loop>
        <loop number="5" name="MWPC_A500_LOOP5">
          <boards>
            <board type="M10" name="MWPC_M10_1" address="9" />
          </boards>
        </loop>
                                </loops>
                        </loopcontroller>
              </loopcontrollers>
              <motionloops>
    <motionloop name="MOTION_LOOP1" S100="MWPC_S100_1">
      <motioncontroller name="MOTION_1" address="5"  />
    </motionloop>
    <motionloop name="MOTION_LOOP2" S100="MWPC_S100_2">
      <motioncontroller name="MOTION_2" address="2"  />
      <motioncontroller name="MOTION_3" address="3"  />
    </motionloop>
              </motionloops>
  <actuators>
    <actuator name="Actuator_1" type="standard" N2400="MWPC_N2400_1"       map1="0"/>
    <actuator name="Actuator_2" type="dual" N2400="MWPC_N2400_1" map1="1"           map2="2" />
  </actuators>
</system>
```

MWPCSystem.xml

4.2 Hardware Reference

4.2.1 Communications and Control (A500)

The A500 provides communication management and processing power to the control application. The Analog Devices 21160M SHARC Digital Signal

Processor (DSP), capable of floating point operation at up to 480MFlops, is the core processor for the board.

Five fiber-optic transmitter/receiver pairs are used to communicate with remote devices that can be arranged in loop configurations. An FPGA is used to fully drive the communications at 10MB/channel, giving a total I/O bandwidth of 50MB/s.

A 100T Ethernet connection allows the A500 to be connected to one or more remote host computers. Sustained output data rates using the UDP protocol are supported to 30MB/s.

Other features included with the A500 include on board flash memory for application program storage, which can be downloaded over the Ethernet connection from the host. A battery-backed up NVR chip allows the application to store critical parameters during run time. One megaword (32 bit) of external SRAM is available for application programming.

A fiber-optic mezzanine allows 10 additional fiber ports to be added. The /R10 adds 10 receivers. The /R5T5 (used by the beamline controller) adds 5 additional fiber loops.

A special I/O mezzanine is present that allows custom I/O boards to be designed and directly connected to the A500. This would be used for high-performance I/O applications, such as might be required for closed loop beam position servo control.

4.2.2 Current Measurement (I3200, I200)

Three products are available to measure low current signals. The I3200 contains 32 independent current integrator channels and the I200 has 2 channels. Both devices can include an additional high-voltage power supply. Each of these devices has two full-scale ranges, 100 pC and 1,000 pC. For a given range, the measured signal is proportional to the integration time, so that very small currents can be read if enough integration time is available. The more sensitive range yields a full-scale signal at 100 nA after one millisecond. A 16-bit digitizer is used so signals that are a small fraction of full scale can be measured. An external high voltage power supply option (HVPS) is also available.

4.2.3 General I/O (M10)

M10 The M10 is suited for control of a single device. The basic configuration includes two channels of 16-bit analog command (+/-10V), two channels of 16-bit analog readback (+/-10V), four channels of TTL digital readback, and four channels of TTL digital command. Used as a single device on a loop, it is capable of 10 kHz update rates on all channels.