



DP4/PX4/DP5 Digital Pulse Processor API Reference

**Amptek, Inc.
14 De Angelo Dr.
Bedford MA 01730**

781-275-2242

www.amptek.com

1 DP4/PX4/DP5 Digital Pulse Processor API (DPPAPI) Basics.....	5
1.1 Programming Guidelines.....	5
1.1.1 Accessing the API.....	5
1.1.2 Establishing Communications.....	5
1.1.3 Selecting a DPP from Multiple Devices.....	5
1.1.4 The DPPAPI Device Index.....	6
1.1.5 DPP Status.....	6
1.1.6 DPP Configuration.....	6
1.1.7 DP5 Configuration Notes.....	6
1.1.8 DP5 Boot Options.....	6
1.1.9 Acquiring Data.....	7
1.1.10 DPPAPI Programming Summary.....	7
1.2 C++ Data Acquisition Example	8
1.3 VB Data Acquisition Example	8
2 DPPAPI Programming Topics.....	9
2.1 Rapid Prototyping Solutions.....	9
2.2 USB.....	9
3 Configurations.....	10
3.1 Reading and Writing DPP Device Configurations.....	10
3.1.1 The DP4 Configuration Cycle.....	10
3.1.2 The PX4 Configuration Cycle.....	10
3.1.3 The DP5 Configuration Cycle.....	11
3.1.4 DP5 Configuration Notes.....	11
3.1.5 DPPAPI Configuration Details.....	11
3.1.6 DPPAPI Active and Temporary Configurations.....	11
3.1.7 Editing Configurations and the CurrentUpdate Parameter.....	12

3.2 Configuration Transfer Functions Diagram.....	12
4 DPP Status Operation.....	13
4.1 DPP Status Function Guide.....	13
4.1.1 Getting Complete Status with One Function Call.....	13
4.1.2 Returning Display Status with One Function Call.....	13
4.1.3 Advanced Status Functions.....	13
4.1.4 Creating Status Information for Amptek Spectrum Files.....	13
4.2 DPP Status Information Types.....	14
4.2.1 Device Identification.....	14
4.2.2 Spectrum Accumulation Status.....	14
4.2.3 Monitors.....	14
4.2.4 Boot Status.....	15
4.2.5 Tuning Status.....	15
5 Acquisition Modes.....	15
5.1.1 Multichannel Analyzer (MCA).....	15
5.1.2 Acquiring MCA Data.....	15
5.1.3 Multichannel Scaling (MCS).....	16
5.1.4 MCS Acquisition Setup.....	16
5.1.5 Acquiring MCS Data.....	16
5.2 Presets.....	17
5.2.1 Preset Modes.....	17
5.2.2 None (No Presets).....	17
5.2.3 Preset Time.....	17
5.2.4 Preset Counts.....	17
5.2.5 Preset Time and Preset Counts.....	17
5.3 Setting the Slow and Fast Thresholds.....	18
6 Automating the DPP Tuning Process.....	19
6.1 Fast Threshold.....	19
6.2 The Tune Fast Threshold Procedure.....	19
6.3 Input Offset.....	19
6.4 Slow Threshold.....	19
7 Configuration Settings.....	20
7.1 MCA Properties.....	20
7.1.1 Acquisition Mode.....	20
7.1.2 MCS Timebase.....	20
7.1.3 MCA Channels.....	21
7.1.4 Buffer Select (DP4/PX4).....	21

7.1.5 TTL Gate.....	21
7.1.6 Slow Threshold.....	21
7.1.7 Preset Time.....	22
7.1.8 Preset Count.....	22
7.1.9 Single Channel Analyzers.....	22
7.2 Shaping Properties.....	23
7.2.1 Time to Peak.....	23
7.2.2 Decimation.....	24
7.2.3 Flat Top.....	24
7.2.4 PUR Enable.....	24
7.2.5 Fast Threshold.....	24
7.2.6 RTD Slow (DP4/PX4) / RTD Ratio (DP5).....	24
7.2.7 RTD On.....	25
7.2.8 RTD Fast (DP4/PX4) / RTD Slow Threshold (DP5).....	25
7.2.9 Baseline Restoration (BLR).....	25
7.2.10 Baseline On (Obsolete, do not use , kept for backward compatibility).....	27
7.3 Gain PZ Properties.....	27
7.3.1 Coarse Gain.....	27
7.3.2 Fine Gain.....	28
7.3.3 Pole Zero (PX4/DP5 only).....	28
7.3.4 Input Offset (PX4/DP5 only).....	28
7.3.5 Input Polarity (PX4/DP5 only).....	29
7.3.6 Detector Reset Lockout Period.....	29
7.4 Power Properties.....	29
7.4.1 Thermoelectric Cooler (PX4/DP5 only).....	29
7.4.2 High Voltage Enabled (PX4/DP5 only).....	29
7.4.3 High Voltage (PX4/DP5 only).....	30
7.4.4 Preamp Voltage (PX4 only).....	30
7.5 Miscellaneous Properties.....	30
7.5.1 Analog Out.....	30
7.5.2 Output Offset.....	30
7.5.3 Auxiliary Output (PX4/DP5 only).....	31
7.5.4 Audible Counter (PX4 only).....	31
8 DPPAPI Data Types.....	32
8.1 DPP_CONFIG_SETTINGS Data Type (C++).....	32
8.2 DPP_CONFIG_SETTINGS Data Type (Visual Basic).....	33
8.3 DPP_STATUS Data Type (C++).....	34

8.4 DPP_STATUS Data Type (Visual Basic).....	34
9 Amptek DPP Configuration File Reference.....	35

1 DP4/PX4/DP5 DIGITAL PULSE PROCESSOR API (DPPAPI) BASICS

The DP4/PX4/DP5 Digital Pulse Processor API is an Application Programming Interface Library for Amptek DP4, PX4, and DP5 Digital Pulse Processors. **The DP4, PX4, and DP5 will be referred to collectively as DPP (Digital Pulse Processor).** The DPPAPI communicates to the DPP via USB. The DPPAPI can select a DPP from a number of DPP devices attached to a system. The DPPAPI runs on personal computers with Intel® Pentium® or higher equivalent processor, Microsoft Windows® 98 (SE), Me, 2000, or XP and 64 MB RAM (128 MB recommended).

Application programs invoke DPPAPI functions to establish DPP communications, control DPP operations, and retrieve spectrum data. The DPPAPI is distributed with an import library (.LIB), and a dynamic link library (.DLL) and a type library (.TLB). Applications can be developed with the DPPAPI in C++, Visual Basic and many other programming languages.

The C language representation of DPPAPI functions are defined in the DppApi.h header file. These files are provided with the DPPAPI libraries. The USB drivers provided with your Amptek DPP must be installed and the usbdrvd.dll must be accessible to the DPPAPI in order to function properly.

1.1 PROGRAMMING GUIDELINES

1.1.1 Accessing the API

Before any interaction with the DPP can take place, the application program must open an instance of the DPPAPI by calling the OpenDppApi function. The OpenDppApi function call creates a handle to the DPPAPI which is then passed to all DPPAPI functions.

After all DPP operations have completed and before the application is terminated the DPPAPI must be closed by calling the CloseDppApi function. The application should always call CloseDppApi before exiting the program.

1.1.2 Establishing Communications

The DPPAPI supports USB communications for one DPP device (called a "connection") from a pool of one or more devices. Before configuring or acquiring data, USB communications must be established. This is done by calling OpenUSBDevice. OpenUSBDevice returns the number of the devices detected. If the number of devices is zero, the DPP cannot be detected. Check if the DPP is ON and the USB cable is plug into the computer. Once a connection is opened, MonitorUSBDevice can be used to check the USB connection. When communications are done call CloseUSBDevice to close the USB connection. CloseUSBDevice must be called before CloseDppApi is called.

1.1.3 Selecting a DPP from Multiple Devices

To select a DPP from a pool of DPP devices, the devices first must be enumerated. The enumeration process begins with a call to OpenUSBDevice to count the number of DPP devices attached. The number of devices is saved and the connection is closed (CloseUSBDevice).

Each DPP must then be identified. The device index ranges from 1 to the number of DPP devices. Each device index is matched to the device type and serial number. OpenUSBDeviceEx opens a device connection for a specific index. GetUSBDeviceInfo returns the DPP device type and DPP serial number for the opened device. CloseUSBDevice must be called before another device is opened and before CloseDppApi is called.

1.1.4 The DPPAPI Device Index

The DPPAPI stores a copy of the DPP device index to determine which USB device to communicate to. Only two functions can change this value, `OpenDppApi` and `OpenUSBDeviceEx`. The DPPAPI device index defaults to one when the DPPAPI is first opened (`OpenDppApi`). Calls to `OpenUSBDeviceEx` changes the DPPAPI device index to the selected device.

`OpenUSBDevice` always [opens the first device if only one device is used](#) and [opens the last index called by `OpenUSBDeviceEx` otherwise](#). The DPPAPI device index can be reset by reopening the DPPAPI or calling `OpenUSBDeviceEx` with an index of one.

1.1.5 DPP Status

The status can be read in a number of formats. Status formats include *display string*, *byte block*, and *status structure*. DPP status includes identification, configuration, monitor, and acquisition information. `GetStatusStruct` returns the status in a structure. Status returned in a status structure can be tested and displayed. Example functions are provided demonstrating how to check the DPP status.

1.1.6 DPP Configuration

The DPP and the DPPAPI must each have a working configuration to perform data acquisition. The DPP and the DPPAPI can get their configurations from different sources. The simplest example is a PX4 that has been configured at power-up by holding the power button down until the device beeps twice. The PX4 configuration has been loaded at power-up, so the only remaining configuration task is to get the configuration being used from the PX4 by calling `GetConfigFromDpp`. This copies the configuration stored in the PX4 hardware and loads the configuration into the DPPAPI program. Even though the PX4 is configured, the DPPAPI must have the configuration to properly acquire data and perform configuration management.

1.1.7 DP5 Configuration Notes

The DP5 has additional features that add more functionality. The [FPGA Clock \(DP5 ONLY\)](#) allows for more control over peaking time. The DP5 FPGA 80MHz Clock Mode indicator must be set to correctly interpret DP5 configuration file settings. The FPGA 80MHz Clock indicator is set when a DPP status command is issued and a DPP device is attached or when `SetFPGAClockDefault` sets this value. **`SetFPGAClockDefault` is needed when editing DP5 configuration files without a DP5 device.** `Get80MHzMode` reads the DP5 FPGA 80MHz Clock Mode indicator. The DP5 FPGA Clock setting has 2 speeds, 20MHz (normal speed = 1) and 80MHz (high speed = 1). Always read the status before configuring a DP5. **`Run GetStatusStruct or GetStatusString before running Get80MHzMode.`**

1.1.8 DP5 Boot Options

DP5 Boot Options allow new configuration options not available with either the DP4 or the PX4. `GetBootOptionsStruct` reads the DP5 Boot Options. DP5 Boot Options can be changed calling `SendDppVendorRequest` with corresponding Vendor Request values.

1.1.9 Acquiring Data

Data can be acquired after the DPP and the DPPAPI have been configured. If the DPP is already acquiring data the only call necessary to acquire data is GetDppData. The data is stored in a buffer ready to be plotted. GetDppData returns the number of channels represented in the spectrum data. Data acquisition can be enabled and paused by calling the PauseDppData function. The data channels are cleared by calling ClearDppData. The data returned by GetDppData function is a summary of the DPP histogram memory: The histogram memory operates as in a traditional MCA. When a pulse occurs with a particular peak value, a counter in a corresponding memory location is incremented. The result is a histogram, an array containing, in each cell, the number of events with the corresponding peak value. This is the energy spectrum and is the primary output of the DPP. The DPP uses 3 bytes per channel, which allows up to 16.7M counts per channel. The 3 bytes per channel are combined producing the final the energy spectrum ready to plot.

1.1.10 DPPAPI Programming Summary

The typical tasks required to setup and acquire data can be summarized as follows:

- open the DPPAPI
- open USB communications
- get the device status
- configure the software
- configure the hardware
- pause the DPP data acquisition
- clear the data buffer
- start the data acquisition
- repeat the next 2 steps until data acquisition finished
 - a. get the acquired data buffer
 - b. plot data
- close usb
- close DPPAPI

Each DPPAPI task can be completed by a function call. A timer is needed for the data acquisition. A method to plot the data must also be added.

1.2 C++ DATA ACQUISITION EXAMPLE

```

void CvcMinExDlg::OnBnClickedGetdatabutton() {
    GetDlgItem(IDC_GETDATABUTTON)->EnableWindow(false); // disable this button until done
    int numdev; // number of usb devices
    void * objDppApi; // pointer to DPPAPI
    char szStatus[2000]; // status text
    long DataBuffer[8192]; // spectrum data
    CString cstrStatus; // status display string
    int NumChan; // number of acquired channels
    DPP_STATUS StatusLst; // status structure

    objDppApi = OpenDppApi(); // Create/Open DPPAPI
    numdev = OpenUSBDevice(objDppApi); // Open USB communications
    if (numdev > 0) {
        GetStatusStruct(objDppApi, true, &StatusLst); // get device status structure
        if (StatusLst.SerialNumber < 1) {
            GetDlgItem(IDC_STATUS)->SetWindowText("Device status error.");
        } else if (! StatusLst.SwConfigRcvd) {
            GetDlgItem(IDC_STATUS)->SetWindowText("Please configure device before taking data.");
        } else {
            if (! StatusLst.StatMcaEnabled) {
                PauseDppData(objDppApi, false); // enable data acquisition
            }
            GetStatusString(objDppApi, 1, szStatus, 2000); // get device status display string
            cstrStatus = szStatus;
            GetDlgItem(IDC_STATUS)->SetWindowText(cstrStatus); // display status
            GetConfigFromDpp(objDppApi); // configure from stored configuration
            NumChan = GetDppData(objDppApi, DataBuffer); // acquire data
            PlotData(DataBuffer, NumChan); // plot data
        }
    }
    CloseUSBDevice (objDppApi); // close usb
    CloseDppApi (objDppApi); // close DPPAPI
    GetDlgItem(IDC_GETDATABUTTON)->EnableWindow(true);
}

```

1.3 VB DATA ACQUISITION EXAMPLE

```

Private Sub cmdGetData_Click()
    cmdGetData.Enabled = False ' disable button until done
    Dim numdev As Integer ' number of usb devices
    Dim objDppApi As Long ' pointer to dpp api
    Dim szStatus As String * 2000 ' status text
    Dim DataBuffer(8192) As Long ' spectrum data
    Dim NumChan As Integer ' number of acquired channels

    objDppApi = OpenDppApi() ' create/open DPPAPI
    numdev = OpenUSBDevice(objDppApi) ' open USB communications
    If (numdev > 0) Then
        GetStatusString objDppApi, 1, szStatus, 2000 ' get device status
        txtStatus = szStatus
        GetConfigFromDpp objDppApi ' configure from hardware
        NumChan = GetDppData(objDppApi, VarPtr(DataBuffer(0))) ' acquire data
        PlotData DataBuffer, NumChan ' plot data
    End If
    CloseUSBDevice (objDppApi) ' close usb
    CloseDppApi (objDppApi) ' close DPPAPI
    cmdGetData.Enabled = True
End Sub

```

2 DPPAPI PROGRAMMING TOPICS

2.1 RAPID PROTOTYPING SOLUTIONS

The DPPAPI was designed for fast and simple development. The source code examples included with the DPPAPI demonstrate most of the functionality. Example code can be cut and pasted into applications to save time and development effort.

- Start with the closest model to the required application.
- Borrow from the other examples.
- Complete and test the basic DPP specific tasks.
- Customize the application.

2.2 USB

The DPPAPI USB interface is a wrapper around the APAUSB interface. No special programming is required. The DPPAPI USB interface includes all the functionality to control and communicate with DP4, PX4 and DP5 type devices. Almost all necessary USB functions are completed automatically by the DPPAPI. The only functions required by the programmer are to open and close the USB device and to monitor if the device has been removed or disconnected. For details of the DPPAPI USB functions see the [USB Manager Functions](#) section.

NOTE: The usbdrv.dll APAUSB driver library must be accessible to the DPPAPI.

3 CONFIGURATIONS

3.1 READING AND WRITING DPP DEVICE CONFIGURATIONS

3.1.1 The DP4 Configuration Cycle

The DP4 stores a copy of the current configuration that can be read back **until the unit is powered down**. Each time a DP4 is powered up, it must be sent a valid configuration. Each time a configuration parameter is changed, a copy of the current configuration must be sent to the DP4. It is also recommended that the new configuration be saved to a file for future reference. There are two methods to detect if a DP4 has been configured with a valid configuration. The *Software Configuration Received* boot status bit can be checked, or the configuration can be read back from the DP4 and compared to the configuration sent.

DP4 Start Up Configuration Sequence:

- Load a configuration file (**GetConfigFromFile**)
- Send a configuration to the DP4 (**SendConfigToDPP**)

DP4 Configuration Update Sequence:

- Update the current configuration (**GetTempConfigSettings**, **SetTempConfigSettings**, etc.)
- Send the new configuration to the DP4 (**SendConfigToDPP**)
- Save the current configuration to file (**SaveConfigToFile**)

3.1.2 The PX4 Configuration Cycle

The PX4 stores a copy of the current configuration in EEPROM. It can be recalled at power up by pressing the power button for about three seconds (until two beeps are heard). If the PX4 has been configured by recalling the stored configuration at power up, the *Power Button Configured* boot status bit will be set. Each time a PX4 is powered up, either the current configuration must be recalled or a valid configuration must be sent. Each time a configuration parameter is changed, a copy of the current configuration must be sent to the PX4. It is also recommended that the new configuration be saved to a file for future reference. There are three methods to detect if a PX4 has been configured with a valid configuration. The *Software Configuration Received* and *Power Button Configured* boot status bits can be checked, or the configuration can be read back from the PX4 and compared to the configuration sent.

Px4 Start Up Configuration Sequence, Power Up Configured Method:

- Power up configure PX4
- Read the current configuration from the PX4 (**GetConfigFromDPP**)
- (Optional but recommended) Send a configuration to the PX4 (**SendConfigToDPP**)

Px4 Start Up Configuration Sequence, Configure from File Method:

- Same as the DP4 Start Up Configuration Sequence

Px4 Configuration Update Sequence:

- DP4 Configuration Update Sequence

3.1.3 The DP5 Configuration Cycle

The DP5 stores a copy of the current configuration in EEPROM. It can be recalled at power up by setting the "**Boot configuration state**" boot flag to "**Use nonvolatile configuration**". If the DP5 has been configured by recalling the stored configuration at power up, the *Power Button Configured* boot status bit will be set. Each time a configuration parameter is changed, a copy of the current configuration must be sent to the DP5. It is also recommended that the new configuration be saved to a file for future reference. There are three methods to detect if a DP5 has been configured with a valid configuration. The *Software Configuration Received* and *Power Button Configured* boot status bits can be checked, or the configuration can be read back from the DP5 and compared to the configuration sent. The DP5 configuration sequence is the same as the PX4.

3.1.4 DP5 Configuration Notes

- The FPGA Clock indicator must be set with SetFPGAClockDefault if a configuration file being edited for a DP5 without a DP5 connected.
- Before configuring a DP5 always read the DP5 status. This sets the FPGA Clock indicator.

3.1.5 DPPAPI Configuration Details

The DPPAPI stores a working copy of the current DPP configuration (**active configuration**). After the DPPAPI is opened, a configuration is read from file or transferred from a DPP device. In the case of the DP4, the active configuration is always read from a file. In the case of the PX4, the initial configuration can be read from a file or from PX4 configuration storage memory. If the configuration is changed, it is recommended that a copy be saved to a file for future reference.

The DPPAPI also stores a spare copy of the current DPP configuration (**temporary configuration**). The temporary configuration can be used as a scratchpad for configuration updates. GetTempConfigSettings and SetTempConfigSettings read/write DPP configuration settings from the active configuration and temporary configuration to/from a DPP_CONFIG_SETTINGS structure. The DPP_CONFIG_SETTINGS structure values are designed to be used with dialog controls.

3.1.6 DPPAPI Active and Temporary Configurations

The DPPAPI stores an active and a temporary configuration. The active configuration is the main DPPAPI configuration. The DPPAPI uses the active configuration for configuration transfer operations. Configuration transfer operations include reading/writing configuration files and sending/receiving DPP device configurations. The temporary configuration is used for configuration editing. When ever a DPPAPI configuration is edited, a copy of the temporary is used.

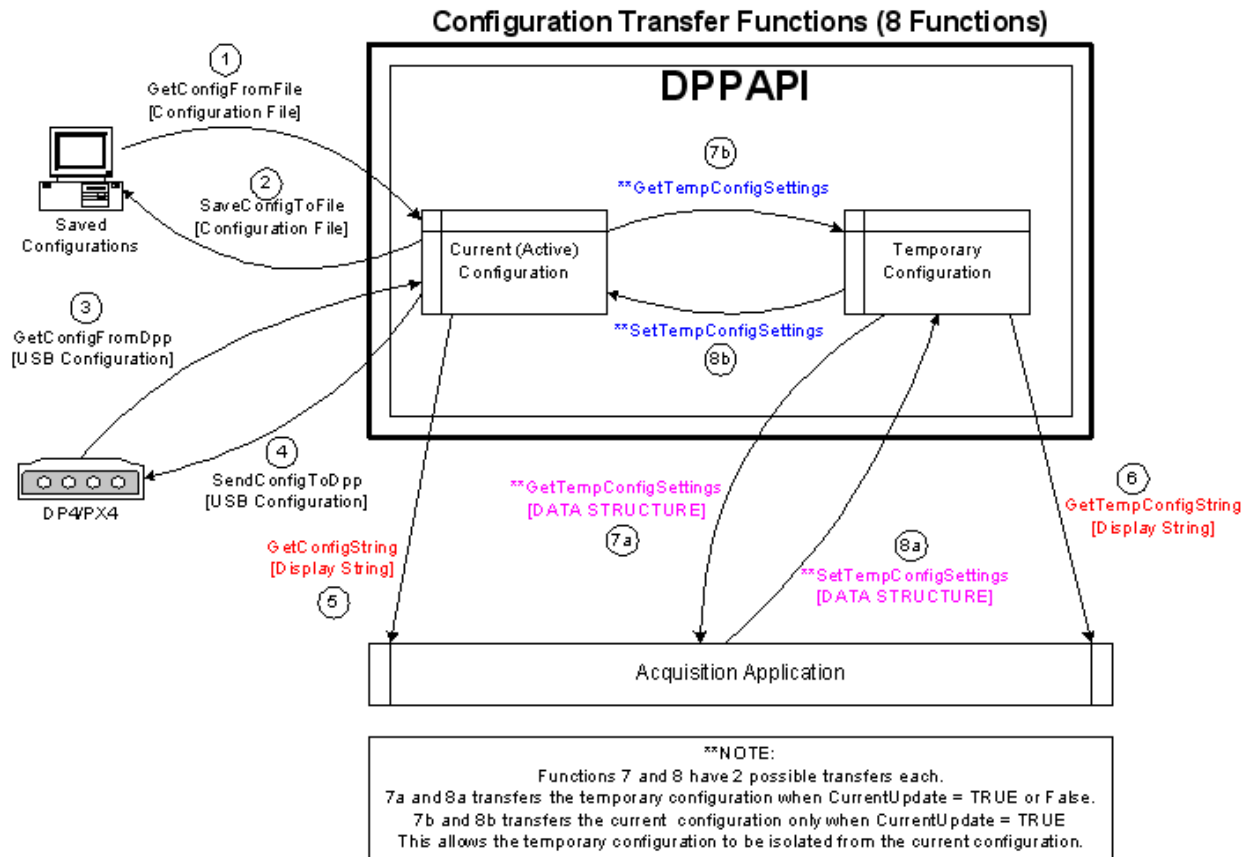
The CurrentUpdate parameter in GetTempConfigSettings and SetTempConfigSettings set to "1 (TRUE)" causes the active configuration settings to be updated along with the temporary settings. This allows the active configuration to be read and directly updated.

The CurrentUpdate parameter in GetTempConfigSettings and SetTempConfigSettings set to "0 (FALSE)" causes the active configuration settings to be isolated from the temporary settings. This allows the temporary configuration to be read and updated independently from the active configuration.

3.1.7 Editing Configurations and the CurrentUpdate Parameter

Set the CurrentUpdate parameter in GetTempConfigSettings and SetTempConfigSettings to "1 (TRUE)" to read and directly update the active configuration.

3.2 CONFIGURATION TRANSFER FUNCTIONS DIAGRAM



4 DPP STATUS OPERATION

DPP Status provides device information for normal operations. Status information can be read as a block of data, a status structure or a display string. Status information includes device identification, spectrum accumulation status, monitors, and boot status. Also see the DPPAPI STATUS INDICATOR FUNCTIONS section.

4.1 DPP STATUS FUNCTION GUIDE

4.1.1 Getting Complete Status with One Function Call

Only one function call is required to get a full snapshot of the current status. Status can be read as a structure with **GetStatusStruct**. In a structure (see DPP_STATUS) the status values are represented in numeric form. This format allows for the values to be easily tested. GetStatusStruct is the preferred method of reading the status. Values in the DPP_STATUS structure can also easily be converted into strings for display.

4.1.2 Returning Display Status with One Function Call

GetStatusString returns a subset of the status for display purposes.

GetStatusString Display

Device Type = PX4
Fast Count = 660
Slow Count = 495
FPGA = 4.00
Firmware = 4.00
Serial Number = 1070
Accumulation Time = 72
StatMcaEnabled = True

4.1.3 Advanced Status Functions

GetStatusBuffer *only* returns the status as a block of bytes (raw data). **Bytes within the data block must be decoded.** The data can be manually decoded (see the PX4 Programmer's Guide) or **ProcessStatusBuffer** to decode the status values. If **ProcessStatusBuffer** is called, the decoded status information is stored in the DPPAPI. A call to **DisplayStatusBuffer** converts the status information into a text display string which is the same format as the GetStatusString Display string.

4.1.4 Creating Status Information for Amptek Spectrum Files

CreateMCAFileDPPSettings reads the current status and the current configuration stored in the DPPAPI. The status and configuration is stored in string format. This function is used to create Amptek DPP MCA Spectrum file configuration and status information. (See ADMCA Help - Amptek Spectrum File Format, vbDppApi example application.

The current status and configuration are updated during normal operations. It is important to understand **how** the current information is updated.

The **current configuration** is updated when a file is read into the DPPAPI or when the current device settings are read into the DPPAPI.

The **current status** is updated when one of the following events occurs:

- **GetStatusStruct** is called
- **GetStatusString** is called.
- **GetStatusBuffer** together with **ProcessStatusBuffer** is called.

4.2 DPP STATUS INFORMATION TYPES

4.2.1 Device Identification

4.2.1.1 Serial Number

The serial number identifies the individual unit. The DPPAPI also uses the serial number to determine if communications are present between the DPP and the DPPAPI.

4.2.1.2 FPGA Version

The Field Programmable Gate Array (FPGA) version identifies the current hardware capabilities. The FPGA is an array of logic gates that is hardware-programmed to fulfill DPP-specified tasks.

4.2.1.3 Firmware Version

The firmware version identifies the low level software that controls the system hardware. The firmware is a set of instructions and data programmed into the EEPROM responsible for controlling the operation of the DPP device. The DPP firmware version value is composed of two parts, a major version and a version. A major version of 4 indicates a DP4 or PX4. A major version of 5 indicates a DP5.

4.2.2 Spectrum Accumulation Status

4.2.2.1 Accumulation Time

Accumulation Time is the duration (real time) of the present data acquisition interval.

4.2.2.2 Fast Count

Fast Count is the number of counts that have accumulated in the fast channel in the present data acquisition interval.

4.2.2.3 Slow Count

Slow Count is the number of counts that have accumulated in the slow channel in the present data acquisition interval.

4.2.3 Monitors

4.2.3.1 Board Temperature

Board Temperature is the current temperature measured on the printed circuit board in Celsius degrees.

4.2.3.2 High Voltage (PX4/DP5 only)

High Voltage is the detector high voltage measured in volts.

4.2.3.3 Thermoelectric Cooler (PX4/DP5 only)

Thermoelectric Cooler is the detector temperature measured in Kelvin.

4.2.4 Boot Status

4.2.4.1 Power Button Configured (PX4 only)

The Power Button Configured bit indicates a PX4 has been configured using the power button.

4.2.4.2 Software Configuration Received

The Software Configuration Received bit indicates a valid configuration has been received since power up.

4.2.4.3 Preset Count Expired (PX4/DP5 only)

The Preset Count Expired bit indicates the preset counts set in the selected range have been reached.

4.2.4.4 Status MCA Enabled

The status MCA enabled bit set indicates the DPP device multichannel analyzer is currently acquiring spectrum data. The DPPAPI makes every effort to preserve data integrity during data acquisition. The MCA acquisition mode is set by the PauseDPPData command.

4.2.4.5 Status Device Indicator (0=DP4, 1=PX4)

The status device indicator identifies the type of DPP device connected. Application features is automatically enabled or disabled depending upon the type device attached. For the DP5, a status device indicator 0="DP4 emulation mode" is selected and 1="PX4 emulation mode" is selected.

4.2.5 Tuning Status

4.2.5.1 Set Fast Threshold Done

DPP tune fast threshold has successfully completed.

4.2.5.2 Set Input Offset Done (PX4/DP5 only)

PX4 tune input offset has successfully completed.

5 ACQUISITION MODES

A primary function of the DPPAPI is to transmit the spectrum to the user. The interface also controls data acquisition, by starting and stopping the processing and by clearing the spectrum data. There are two acquisition modes **Multichannel Analyzer (MCA)** and **Multichannel Scaler (MCS)**. For a detailed description of MCA and MCS modes see the DP4 and PX4 Users Manuals. All the configuration settings play an important role in acquiring data in MCA and MCS modes.

5.1.1 Multichannel Analyzer (MCA)

The Multichannel Analyzer (MCA) produces an energy spectrum. The digitized peak amplitude of an input shaped pulse creates a count. Counts are accumulated into channels representing a narrow range of energy. The histogram of the counts is the energy spectrum. This is the normal mode of operation.

5.1.2 Acquiring MCA Data

Before starting an acquisition, all configuration settings must be correctly set. The "Acquisition Mode" configuration setting (AcqMode) must be set to MCA. GetDppData returns a snapshot of the data buffer ready for histogram display or for mca plot file storage. The status indicator StatMcaEnabled set to TRUE indicates the acquisition device MCA is enabled. When MCA is paused StatMcaEnabled is false.

5.1.3 Multichannel Scaling (MCS)

The MultiChannel Scaler (MCS) is similar to the MCA except that each data channel corresponds to time instead of amplitude (energy), with the channel boundaries corresponding to start and stop times. Each value in the histogram array represents the total number of counts above the slow threshold during the appropriate time interval.

An MCS data acquisition is started with a given counting time per channel (MCS Timebase). The system records the number of counts in the first interval and places them in the first channel, and so on. The slow (shaped) channel counts are used for MCS. If the peaking time is set to its fastest (0.8 ms), then the MCS counts more closely resemble the input count rate. The MCS mode uses SCA8, so this gives the user the capability of setting the upper and lower level thresholds

5.1.4 MCS Acquisition Setup

The region of interest (energy range) is set using SCA8. Set SCA8 Low Channel to "0" and SCA8 High Channel to "**MCA Channels – 1**". "**0 to MCA Channels – 1**" is the maximum range and is used for normal operation. **SCA8 MUST BE ENABLED AND SET FOR MCS TO RUN.**

5.1.5 Acquiring MCS Data

Configure the acquisition device and take a sample acquisition in MCA mode. This ensures the acquisition device and the DPPAPI are functioning and ready. The next step is to setup the MCS related parameters. Set the Acquisition Mode to MCS. Select a MCS Timebase. Set the energy range in SCA8 (From 0 to MCA Channels - 1). Enable SCA8 and apply the configuration. Once the MCS parameters are set, the DPP and DPPAPI are ready for MCS data acquisition. Stop the acquisition, if running. Clear the current data. Start the MCS acquisition. The status indicator MCSDone set to TRUE indicates the MCS acquisition has completed. The spectrum data is read and saved the same as in the MCA mode.

5.2 PRESETS

Presets are set as part of the DPP configuration. There are two types of presets, **Preset Time** and **Preset Counts**. The preset time and preset count values are stored in separate variables. Preset counts also requires channel boundary values and an enable which are stored in SCA8.

5.2.1 Preset Modes

There are four possible preset modes:

- None. – No presets are set.
- Preset Time – A preset time is set, but no preset count is set.
- Preset Count – A preset count is set, but no preset time is set.
- Preset Time and Count – Both a preset time and count is set.

5.2.2 None (No Presets)

To turn off all presets, the Preset Time must be set to zero or none and the preset counts must be disabled. To ensure that the preset counts are disabled, set the preset counts to zero, set the lower and upper preset channels to zero and set the SCA8 enable to disabled.

5.2.3 Preset Time

Preset Time stops the spectrum acquisition at a predetermined time.

- If the preset time value is set before the spectrum acquisition has begun then the spectrum acquisition will stop when the Accumulation Time reaches the preset time.
- If the preset time value is set after the spectrum acquisition has begun and the Accumulation Time is less than the time set, the spectrum acquisition will stop when the Accumulation Time reaches the preset time.
- If the preset time value is set after the spectrum acquisition has begun and the Accumulation Time is greater than the time set, the spectrum acquisition will not stop until after a new acquisition has begun and the Accumulation Time reaches the preset time.

5.2.4 Preset Counts

Preset Count stops the spectrum acquisition at a predetermined number of counts for a selected group of channels. Select the low and upper count boundary channels and enable SCA8. The preset counts are stored in a variable and the selected channels and the SCA8 enable are stored in SCA8.

5.2.5 Preset Time and Preset Counts

When both **Preset Time** and **Preset Counts** are selected the acquisition stops when the first preset is reached.

5.3 SETTING THE SLOW AND FAST THRESHOLDS

Setting the fast and slow thresholds is an important setup consideration. Following is a step-by-step guide to correctly setting the fast and slow thresholds.

- 1) Configure the DPP and the application software for an acquisition.
- 2) Start an acquisition.
- 3) Remove the source.
- 4) Set to delta mode.
- 5) Setting the slow threshold:
 - A) Look for counts on low end.
 - B) Place cursor above low end counts.
 - C) Open configuration dialog.
 - D) Go to MCA tab.
 - E) Press slow threshold cursor button.
 - F) Press apply.
- 6) Setting the fast threshold:
 - A) Open configuration dialog.
 - B) Go to Shaping Tab
 - C) Select a fast threshold.
 - D) Press apply.
 - E) Check Fast Count value in Info Panel.
 - F) Repeat Steps C-E until the Input Counts value is approximately 5.
- 7) The thresholds are now set. You are ready to begin an acquisition.
- 8) Replace the source.
- 9) Start an acquisition.

6 AUTOMATING THE DPP TUNING PROCESS

6.1 FAST THRESHOLD

Calling the **TuneFastThreshold** function automatically tunes the fast channel threshold. Please wait until the operation has completed before proceeding.

6.2 THE TUNE FAST THRESHOLD PROCEDURE

1. **REMOVE THE SOURCE BEFORE RUNNING.**
2. Press Tune Fast Threshold button.
3. The dialog may not be visible for a second.
4. Wait for the dialog box to return.
5. The old fast threshold is replaced with the new value.
6. Tune Fast Threshold only takes a few seconds to run.

6.3 INPUT OFFSET

Calling the **TuneInputOffset** function (PX4/DP5) automatically sets the input signal offset. First a coarse input offset adjust is done, then a fine input offset is done to complete the input offset adjustment.

6.4 SLOW THRESHOLD

A simple algorithm to tune the slow threshold:

1. **REMOVE THE SOURCE BEFORE RUNNING.**
2. Set the slow threshold to 0.
3. Start an acquisition.
4. Clear the spectrum data.
5. Wait a few seconds.
6. Stop the acquisition.
7. Save the spectrum data to a buffer.
8. Search the buffer at the low channel end for slow counts.
9. Set the slow threshold above the highest slow counts channel.
10. Adjust for the specific application.

7 CONFIGURATION SETTINGS

The DPP_CONFIG_SETTINGS data structure variables share the same data type in C++ and Visual Basic. The DPP_CONFIG_SETTINGS data values have been selected for ease of use with application controls.

7.1 MCA PROPERTIES

7.1.1 Acquisition Mode

Variable	AcqMode
Data Type	byte
Description	The acquisition mode selection determines how the acquired events will be analyzed. The standard acquisition mode is MCA or <i>Multi-Channel Analyzer</i> mode. Amptek Digital Pulse Processors are designed to be <i>Multi-Channel Analyzers</i> . <i>Multi-Channel Analyzers</i> create a histogram of pulse heights over a particular voltage scale.

As added functionality, a limited MCS or *Multi-Channel Scaling* mode has been added. Once per update interval, the data from all the channels is downloaded and summed. This summed value is then plotted in the display. Therefore, what is displayed is the count rate per update interval.

Values	0 - MCA
	1 - MCS

7.1.2 MCS Timebase

Variable	MCSTimebase
Data Type	byte
Description	Sets the MCS time interval per channel.

Values

0 - 10mS/channel	8 - 5 sec/channel
1 - 20mS/channel	9 - 10 sec/channel
2 - 50mS/channel	10 - 20 sec/channel
3 - 100mS/channel	11 - 30 sec/channel
4 - 200mS/channel	12 - 60 sec/channel
5 - 500mS/channel	13 - 90 sec/channel
6 - 1 sec/channel	14 - 120 sec/channel
7 - 2 sec/channel	15 - 300 sec/channel

7.1.3 MCA Channels

Variable	MCAChannels
Data Type	byte
Description	Number of acquisition channels.
Values	0 - 256 1 - 512 2 - 1024 3 - 2048 4 - 4096 5 - 8192

7.1.4 Buffer Select (DP4/PX4)

Variable	BufferSelect
Data Type	byte
Description	MCA buffer selection, the selected MCA buffer holds the current accumulating counts.
Values	DP4/PX4: 0 - Buffer A 1 - Buffer B 2 - HW Select DP5: 0 - Buffer A (DP5 Supports Buffer A Only)

7.1.5 TTL Gate

Variable	TTLGate
Data Type	byte
Description	TTL gate input includes or excludes events from the spectrum.
Values	0 - Off 1 - Active High 2 - Active Low

7.1.6 Slow Threshold

Variable	SlowThreshold
Data Type	byte

Description	Pulses exceeding the low level discriminator threshold on the slow channel will be recorded in the spectrum.
Values	0 – 250 = 0 - 24.41% of the full energy scale in percent or channels

7.1.7 Preset Time

Variable	PresetTime
Data Type	long
Description	Preset duration of time requested for the current acquisition in tenths of seconds
Values	100ms to 19.4 days, with 100ms precision (i.e. 100ms = 1, 5 seconds = 50, etc.) Control value strings used with configuration helper functions:

0 - None	8 - 5 min
1 - 1 sec	9 - 10 min
2 - 2 sec	10 - 30 min
3 - 5 sec	11 - 1 hr
4 - 10 sec	12 - 2 hr
5 - 30 sec	13 - 5 hr
6 - 1 min	14 - 10 hr
7 - 2 min	15 - 24 hr

7.1.8 Preset Count

Variable	PresetCount
Data Type	long
Description	Preset number of counts requested in selected channel range. Channel values are stored in SCA8.
Values	0 - 0xFFFFFFFF counts

7.1.9 Single Channel Analyzers

Variable	SCA(0 To 7)
Data Type	long
Description	Single channel analyzers, upper/lower thresholds and enables, stored in packed format.
Values	0 - SCA 1 Controls

- 1 - SCA 2 Controls
- 2 - SCA 3 Controls
- 3 - SCA 4 Controls
- 4 - SCA 5 Controls
- 5 - SCA 6 Controls
- 6 - SCA 7 Controls
- 7 - SCA 8 Controls

SCA values, LL, UL, and Enable are stored together

Select Single Channel Analyzer Value Composition:

Description	Enable Channel	Upper Level Threshold	Lower Level Threshold
Bits	31	30-16	15-0
Value	0=OFF, 1=ON	upper channel value – 1=0-8191	lower channel value -1=0-8191

7.2 SHAPING PROPERTIES

7.2.1 Time to Peak

Variable TimeToPeak

Data Type byte

Description Time-to-peak register setting. **For DP5 80MHz FPGA Clock the Peaking Time values are 4x faster.** (These values are multiplies by 0.25.)

Values Configuration helper functions convert these values to configuration values.

Spin - Display string values:

0 - 0.8uS	12 - 16.0uS
1 - 1.6uS	13 - 19.2uS
2 - 2.4uS	14 - 22.4uS
3 - 3.2uS	15 - 25.6uS
4 - 4.0uS	16 - 32.0uS
5 - 4.8uS	17 - 38.4uS
6 - 5.6uS	18 - 44.8uS
7 - 6.4uS	19 - 51.2uS
8 - 8.0uS	20 - 64.0uS
9 - 9.6uS	21 - 76.8uS
10 - 11.2uS	22 - 89.6uS
11 - 12.8uS	23 - 102.4uS

7.2.2 Decimation

Variable	Decimation
Data Type	byte
Description	decimation setting for pulse shaping
Values	Calculated by configuration helper functions from time-to-peak value.

7.2.3 Flat Top

Variable	FlatTop
Data Type	byte
Description	Flattop register setting. For DP5 80MHz FPGA Clock the Flattop Width values are 4x faster. (These values are multiplies by 0.25.)
Values	Display values calculated by configuration helper functions from time-to-peak value and decimation value. Flattop calculated from selected control value.

7.2.4 PUR Enable

Variable	PUREnable
Data Type	byte
Description	pile-up rejection enabled
Values	0 - Off 1 - On

7.2.5 Fast Threshold

Variable	FastThreshold
Data Type	byte
Description	Fast Ch Threshold, events with fast channel amplitude below this are rejected.
Values	0 - 250

7.2.6 RTD Slow (DP4/PX4) / RTD Ratio (DP5)

Variable	RTDSlow
Data Type	byte
Description	DP4/PX4: Risetime Discrimination slow threshold DP5: If RTD is enabled, compares the fast channel peak height to the slow channel peak height. If this ratio is sufficiently high (fast risetime), the event is accepted. If this ratio is low, the event is rejected.
Values	spin values: 0 – 255 display values: 0 – 48.83% full energy scale

7.2.7 RTD On

Variable	RTDOn
Data Type	byte
Description	Turns RTD on, and sets the amplitude and timing thresholds
Values	0 - Off 1 - On

7.2.8 RTD Fast (DP4/PX4) / RTD Slow Threshold (DP5)

Variable	RTDFast
Data Type	byte
Description	DP4/PX4: RTD Time Threshold Events with HWHM wider than this are rejected DP5: If RTD is enabled, sets the RTD slow threshold. This threshold applies to the slow channel. Only events exceeding this threshold will be analyzed by the RTD circuit; all events below this threshold are accepted.
Values	4 - 19

RTD Rules:

Events whose shaped slow channel amplitude is below the RTD slow threshold are kept.

- OR -

Events whose fast channel amplitude is below the fast threshold are rejected;

- OR -

Events whose measured HWHM is wider than the RTD Time Threshold are rejected.

7.2.9 Baseline Restoration (BLR)

Variable	BLR
Data Type	byte
Description	Baseline Restoration, (see VB SpinToBLR function) . For DP5 80MHz FPGA Clock the Baseline Restoration (BLR) values are 4x faster. (See Table of values below.)
Values	

ComboBox Index	BLR Setting	BLR
0	63	BLR:OFF*
1	64	BLR:ON---DOWN:1:VERY SLOW---UP:1:VERY SLOW
2	68	BLR:ON---DOWN:1:VERY SLOW---UP:4:SLOW
3	72	BLR:ON---DOWN:1:VERY SLOW---UP:16:MEDIUM
4	76	BLR:ON---DOWN:1:VERY SLOW---UP:64:FAST
5	80	BLR:ON---DOWN:4:SLOW---UP:1:VERY SLOW
6	84	BLR:ON---DOWN:4:SLOW---UP:4:SLOW

7	88	BLR:ON---DOWN:4:SLOW---UP:16:MEDIUM
8	92	BLR:ON---DOWN:4:SLOW---UP:64:FAST
9	96	BLR:ON---DOWN:16:MEDIUM---UP:1:VERY SLOW
10	100	BLR:ON---DOWN:16:MEDIUM---UP:4:SLOW
11	104	BLR:ON---DOWN:16:MEDIUM---UP:16:MEDIUM
12	108	BLR:ON---DOWN:16:MEDIUM---UP:64:FAST
13	112	BLR:ON---DOWN:64:FAST---UP:1:VERY SLOW
14	116	BLR:ON---DOWN:64:FAST---UP:4:SLOW
15	120	BLR:ON---DOWN:64:FAST---UP:16:MEDIUM
16	124	BLR:ON---DOWN:64:FAST---UP:64:FAST

DP5 80MHz FPGA Clock Only

ComboBox Index	BLR Setting	BLR
0	63	BLR: OFF
1	64	BLR:ON---DOWN:4:SLOW---UP:4:SLOW
2	68	BLR:ON---DOWN:4:SLOW---UP:16:MEDIUM
3	72	BLR:ON---DOWN:4:SLOW---UP:64:FAST
4	76	BLR:ON---DOWN:4:SLOW---UP:256:VERY FAST
5	80	BLR:ON---DOWN:16:MEDIUM---UP:4:SLOW
6	84	BLR:ON---DOWN:16:MEDIUM---UP:16:MEDIUM
7	88	BLR:ON---DOWN:16:MEDIUM---UP:64:FAST
8	92	BLR:ON---DOWN:16:MEDIUM---UP:256:VERY FAST
9	96	BLR:ON---DOWN:64:FAST---UP:4:SLOW
10	100	BLR:ON---DOWN:64:FAST---UP:16:MEDIUM
11	104	BLR:ON---DOWN:64:FAST---UP:64:FAST
12	108	BLR:ON---DOWN:64:FAST---UP:256:VERY FAST
13	112	BLR:ON---DOWN:256:VERY FAST---UP:4:SLOW
14	116	BLR:ON---DOWN:256:VERY FAST---UP:16:MEDIUM
15	120	BLR:ON---DOWN:256:VERY FAST---UP:64:FAST
16	124	BLR:ON---DOWN:256:VERY FAST---UP:256:VERY FAST

7.2.10 Baseline On (Obsolete, do not use , kept for backward compatibility)

Variable BaselineOn
 Data Type byte
 Description use autobaseline during detector reset
 Values 0 - Off

7.3 GAIN PZ PROPERTIES**7.3.1 Coarse Gain**

Variable Coarse Gain
 Data Type byte
 Description coarse gain value, provides large gain adjust
 Values DP4

0 - 10.8x

1 - 20.7x

2 - 55.4x

3 - 106.2x

PX4

0 - 4.13x

14 - 49.93x

1 - 4.95x

15 - 60.30x

2 - 5.94x

16 - 76.70x

3 - 7.17x

17 - 91.86x

4 - 8.22x

18 - 110.29x

5 - 9.84x

19 - 133.19x

6 - 11.82x

29 - 162.92x

7 - 14.27x

21 - 195.10x

8 - 17.46x

22 - 234.25x

9 - 20.90x

23 - 282.89x

10 - 25.10x

24 - 324.14x

11 - 30.31x

25 - 388.17x

12 - 34.73x

26 - 466.05x

13 - 41.59x

27 - 562.83x

DP5	
0 - 1.00x	8 - 14.56x
1 - 2.22x	9 - 17.77x
2 - 3.78x	10 - 22.42x
3 - 5.26x	11 - 30.83x
4 - 6.56x	12 - 38.18x
5 - 8.39x	13 - 47.47x
6 - 10.1x	14 - 66.26x
7 - 11.31x	15 - 102.01x

7.3.2 Fine Gain

Variable Fine Gain
 Data Type double
 Description fine gain value, allows small gain adjustments
 Values Spin values: 6144-10240, fine gain values: 0.7500 – 1.2500

7.3.3 Pole Zero (PX4/DP5 only)

Variable PoleZero
 Data Type long
 Description pole zero adjust value (PX4/DP5 only)
 Values 0 - 250

7.3.4 Input Offset (PX4/DP5 only)

Variable InputOffset
 Data Type long
 Description input offset (PX4/DP5 only)
 Values DP4: 0
 PX4/DP5:
 spin values: 0 – 4000
 display values: -2.048V to 1.952V
 DP5:
 A setting of -2.048V instructs the DP5 to use a DP5 selected default setting, any other setting is interpreted the same as the PX4

7.3.5 Input Polarity (PX4/DP5 only)

Variable	InputPolarity
Data Type	byte
Description	Determines if the PX4 front-end is inverting or not inverting. (PX4/DP5 only)
Values	DP4: 0 PX4/DP5: 0 – Positive = Front-end non-inverting 1 – Negative = Front-end Inverting

7.3.6 Detector Reset Lockout Period

Variable	DetReset
Data Type	byte
Description	Detector reset lockout period. For DP5 80MHz FPGA Clock the Detector Reset Lockout Period values are 4x faster. (These values are multiplies by 0.25.)
Values	0 = Slow - 13.1mS 1 = Slow - 6.55mS 2 = Slow - 3.28mS 3 = Slow - 1.64mS 4 = Fast - 819uS 5 = Fast - 410uS 6 = Fast - 205uS 7 = Fast - 102uS 8 = OFF

7.4 POWER PROPERTIES**7.4.1 Thermoelectric Cooler (PX4/DP5 only)**

Variable	TEC
Data Type	long
Description	TEC temperature setting (displayed in Kelvin). (PX4/DP5 only.)
Values	DP4: Any value within PX4 range. PX4: spin values: 2458 – 4095 display values: 180.0K - 299.9K

7.4.2 High Voltage Enabled (PX4/DP5 only)

Variable	HVEnabled
Data Type	byte

Description high voltage setting enable (PX4/DP5 only)
 Values 0 - Disabled
 1 - Enabled

7.4.3 High Voltage (PX4/DP5 only)

Variable HV
 Data Type long
 Description High voltage setting value. (PX4/DP5 only.)
 Values DP4: Any value within valid PX4 range.
 PX4: spin: 0 – 4000 display values: 0.0V - 2929.7V (only 0V to 1500V is valid)

7.4.4 Preamp Voltage (PX4 only)

Variable PreampPower
 Data Type byte
 Description preamp voltage select value (PX4 only)
 Values 0 - 5.0V
 1 - 8.5V
 (Not recognized by DP5 – preamp voltage set by hardware jumpers)

7.5 MISCELLANEOUS PROPERTIES

7.5.1 Analog Out

Variable AnalogOut
 Data Type byte
 Description Analog output dac enabled and DAC output type.
 Values 0 - Fast Channel
 1 - Shaped Pulse
 2 - Decimated Input
 3 - BLR Correction
 4 – Off

7.5.2 Output Offset

Variable OutputOffset
 Data Type byte
 Description Output DAC offset
 Values spin values: 0 – 127 display values: -500mV to 492mV

7.5.3 Auxiliary Output (PX4/DP5 only)

Variable	AuxOut
Data Type	byte
Description	Auxiliary output select (PX4/DP5 only)
Values	0 - ICR 1 - PILEUP 2 - MCS Timebase 3 - ONESHOT 4 - DET_RES 5 - MCA_EN 6 - TRIGGER 7 - SCA8

7.5.4 Audible Counter (PX4 only)

Variable	AudibleCounter
Data Type	byte
Description	audio volume setting (PX4 only)
Values	0 - Off 1 - Low 2 - Medium 3 - High

8 DPPAPI DATA TYPES

8.1 DPP_CONFIG_SETTINGS DATA TYPE (C++)

```
typedef struct _DPP_CONFIG_SETTINGS {
    //MCA Properties
    byte AcqMode;           // acquisition mode 0=MCA,1=MCS
    byte MCSTimebase;       // MCS timebase value (0-15) see CAcqMode
    byte MCAChannels;       // number of channels 4=256,3=512,2=1024,1=2048,0=4096,5=8192
    byte BufferSelect;       // Holds Buffer Sel A&B,Buffer Sel Hardware,see DPPBufferSelect
    byte TTGate;            // gate input, controls spectrum events included/excluded, see DPPGate
    byte SlowThreshold;     // Slow ch threshold, Events w/amp lower not added to spectrum
    long PresetTime;        // var holds preset time, used in usb
    long PresetCount;       // preset count in selected channels, ch are set in SCA8
    long SCA[8];            // SCA values, LL, UL, and Enable are stored together

    //Shaping Properties
    byte TimeToPeak;        // TimeToPeak register setting
    byte Decimation;        // decimation setting for pulse shaping
    byte FlatTop;           // flatop register setting
    byte PUREnable;         // pile-up rejection enabled
    byte FastThreshold;     // Fast Ch Threshold, events w/fch amp below this are rejected
    byte RTDSlow;           // Risetime Discrimination slow threshold
    byte RTDOn;             // Turns RTD on, and sets the amplitude and timing thresholds
    byte RTDFast;           // RTD Time Threshold,Events w/HWHM wider than this are rejected
    byte BLR;               // Baseline Restoration, see udBLR.Value notes for values
    byte BaselineOn;        // use autobaseline during detector reset

    //Gain PZ Properties
    byte CoarseGain;        // stores current coarse gain value
    long FineGain;          // stores current fine gain value
    long PoleZero;          // pole zero adjust value
    long InputOffset;       // input offset
    byte InputPolarity;     // use InvertEnable during detector reset
    byte DetReset;          // detector reset lockout period

    //Power Properties
    long TEC;               // TEC temperature setting (displayed in Kelvin)
    byte HVEnabled;         // high voltage setting enable
    long HV;                // high voltage setting value
    byte PreamplifierPower; // preamp power select value (5v or 8.5v)

    //Misc Properties
    byte AnalogOut;         // dac enabled and DAC output type, (stobed peak,shaped pulse,dec
    inp,fast ch)
    byte OutputOffset;      // Output DAC offset, -64...+63, (signed) (D7-D1) (-500mV to +492mV)
    byte AuxOut;            // Aux output type
    byte AudibleCounter;    // audio volume setting
} DPP_CONFIG_SETTINGS, *LP_DPP_CONFIG_SETTINGS;
```

8.2 DPP_CONFIG_SETTINGS DATA TYPE (VISUAL BASIC)

```

Public Type DPP_CONFIG_SETTINGS
    ' MCA Properties
    AcqMode As Byte           ' acquisition mode 0=MCA,1=MCS
    MCSTimebase As Byte       ' MCS timebase value (0-15) see CAcqMode
    MCACHannels As Byte       ' number of channels 4=256,3=512,2=1024,1=2048,0=4096,5=8192
    BufferSelect As Byte      ' Holds Buffer Sel A&B,Buffer Sel Hardware,see DPPBufferSelect
    TTLGate As Byte          ' gate input, controls spectrum events included/excluded, see DPPGate
    SlowThreshold As Byte     ' Slow ch threshold, Events w/amp lower not added to spectrum
    PresetTime As Long        ' var holds preset time, used in usb
    PresetCount As Long       ' preset count in selected channels, ch are set in SCA8
    SCA(0 To 7) As Long       ' SCA values, LL, UL, and Enable are stored together

    ' Shaping Properties
    TimeToPeak As Byte        ' TimeToPeak register setting
    Decimation As Byte        ' decimation setting for pulse shaping
    FlatTop As Byte           ' flatop register setting
    PUREnable As Byte         ' pile-up rejection enabled
    FastThreshold As Byte     ' Fast Ch Threshold, events w/fch amp below this are rejected
    RTDSlow As Byte           ' RisetTime Discrimination slow threshold
    RTDOn As Byte             ' Turns RTD on, and sets the amplitude and timing thresholds
    RTDFast As Byte           ' RTD Time Threshold,Events w/HWHM wider than this are rejected
    BLR As Byte               ' Baseline Restoration, see udBLR.Value notes for values
    BaselineOn As Byte        ' use autobaseline during detector reset

    ' Gain PZ Properties
    CoarseGain As Byte        ' stores current coarse gain value
    FineGain As Long          ' stores current fine gain value
    PoleZero As Long          ' pole zero adjust value
    InputOffset As Long        ' input offset
    InputPolarity As Byte     ' use InvertEnable during detector reset
    DetReset As Byte          ' detector reset lockout period

    ' Power Properties
    TEC As Long               ' TEC temperature setting (displayed in Kelvin)
    HVEnabled As Byte         ' high voltage setting enable
    HV As Long                ' high voltage setting value
    Preamplifier As Byte      ' preamp power select value (5v or 8.5v)

    ' Misc Properties
    AnalogOut As Byte         ' DAC enabled/output type, (stobed peak,shaped pulse,dec inp,fast ch)
    OutputOffset As Byte      ' Output DAC offset, -64...+63, (signed) (D7-D1) (-500mV to +492mV)
    AuxOut As Byte            ' Aux output type
    AudibleCounter As Byte    ' audio volume setting
End Type

```

8.3 DPP_STATUS DATA TYPE (C++)

```
typedef struct _DPP_STATUS {
    // Identification
    double FPGA;           // FPGA
    double Firmware;       // firmware revision
    double SerialNumber;   // unit serial number
    byte StatDevInd;       // device indicator from status block (0=dp4,1=px4)
    // Configuration
    byte BootStatus;       // boot status byte (PwrBtnConfig,SwConfigRcvd,McaEnabled)
    byte PwrBtnConfig;     // dpp has loaded config from power button
    byte SwConfigRcvd;     // software has sent valid config to unit since startup
    // Tuning
    byte SetFastThreshDone; // DPP tune fast thresh has successfully done
    byte SetSlowThreshDone; // DPP tune slow thresh has successfully done (NOT USED)
    byte SetInputOffsetDone; // DPP tune input offset has successfully done
    // Monitors
    double BoardTemp;      // board temperature monitor value
    double HVMonitor;      // high voltage monitor value
    double TECMonitor;     // TEC temperature monitor value (displayed in Kelvin)
    // Acquisition Parameters
    double FastCount;      // fast channel count
    double SlowCount;      // slow channel count
    double AccumulationTime; // real time duration of present data acquisition interval
    // Acquisition Mode
    byte StatMcaEnabled;   // mca enabled status, hi during acq, preset time resets flag
    byte MCSDone;         // MCS done flag 1=finished, 0=not finished
    byte PresetCountExpired; // indicates presets counts have been reached
} DPP_STATUS, *LP_DPP_STATUS;
```

8.4 DPP_STATUS DATA TYPE (VISUAL BASIC)

```
Public Type DPP_STATUS
    ' Identification
    FPGA As Double           ' FPGA
    Firmware As Double       ' firmware revision
    SerialNumber As Double   ' unit serial number
    StatDevInd As Byte       ' device indicator from status block (0=dp4,1=px4)
    ' Configuration
    BootStatus As Byte       ' boot status byte (PwrBtnConfig,SwConfigRcvd,McaEnabled)
    PwrBtnConfig As Byte     ' dpp has loaded config from power button
    SwConfigRcvd As Byte     ' software has sent valid config to unit since startup
    ' Tuning
    SetFastThreshDone As Byte ' DPP tune fast thresh has successfully done
    SetSlowThreshDone As Byte ' DPP tune slow thresh has successfully done
    SetInputOffsetDone As Byte ' DPP tune input offset has successfully done
    ' Monitors
    BoardTemp As Double      ' board temperature monitor value
    HVMonitor As Double      ' high voltage monitor value
    TECMonitor As Double     ' TEC temperature monitor value (displayed in Kelvin)
    ' Acquisition Parameters
    FastCount As Double      ' fast channel count
    SlowCount As Double      ' slow channel count
    AccumulationTime As Double ' real time duration of present data acquisition interval
    ' Acquisition Mode
    StatMcaEnabled As Byte   ' mca enabled status, high during acq, preset time resets flag
    MCSDone As Byte         ' MCS done flag 1=finished, 0=not finished
    PresetCountExpired As Byte ' indicates presets counts have been reached
End Type
```

9 AMPTEK DPP CONFIGURATION FILE REFERENCE

The Amptek DPP configuration file format allows for configurations to be shared between DPP devices and software. The current format is a superset of the past configuration file formats. The goal is to provide seamless and uninterrupted development of DPP devices and software. By expanding on the legacy format, development continuity has been maintained with maximum preservation of past development investments. The following table is a summary of the format.

Item	Setting	Stored Value Type	Reference Section	Notes
1	Peaking Time	control index	Time to Peak	This value is used to generate Decimation and TimeToPeak values. The Decimation value is necessary to generate the FlatTop Width display values. These values must be calculated before the FlatTop is set.
2	Flat Top Width	control index	Flat Top AND Decimation	There are 16 Flat Top settings. To calculate the FlatTop Width display value: $\text{index} * 0.1 * (2^{\text{Decimation}})$ The value is displayed in "uS".
3	Slow Threshold	control index	Slow Threshold	
4	Fast Threshold	control index	Fast Threshold	
5	Output Offset	control index	Output Offset	To calculate the Output Offset display value: $i = \text{index} * 2$ $j = \text{If}(i > 127, 1, 0)$ $k = ((i + j * -256) * 1000 / 256)$ OutputOffset=cstring(k)+"mV"
6	PUR	string text	PUR Enable	"PUROff"=0, "PUROn"=1
7	Detector Reset	control index	Detector Reset Lockout Period	
8	Analog Out	string text index	Analog Out	The current control index + 1 is appended to the string "DAC" Examples: index = 0 -> DAC1 index = 1 -> DAC2
9	MCA Channels	string text	MCA Channels	The number or channels is appended to the string "MCA"

				Example: channels = 1024 -> MCA1024
10	Com Port	string text	Only USB supported	Always set to zero (0) or "USB". Developer Notes: ADMCA.exe and DPPAPI.dll currently only support USB and ignore this setting. The VB DPP hardware demo application DP4.exe or PX4.exe uses the following format: USB = "USB" RS232 COM PORT1= "COM1"
11	RTD Enable	string text	RTD Enable	"RTDOff"=0, "RTDOn"=1
12	RTD Slow	control index	RTD Slow	
13	RTD Fast	control index	RTD Fast	
14	Coarse Gain	control index	Coarse Gain	
15	Auto Baseline	string text	Baseline On	"AutoBaselineOff"=0 "AutoBaselineOn"=1
16	BLR	setting value	Baseline Restoration	Note: Use the BLR setting not the index. On some applications these are the same.
17	Buffer Select	string text	Buffer Select	"BufferSelectA"=0 "BufferSelectB"=1 "BufferSelectHW"=2
18	TTL Gate	string text	TTL Gate	"GateOff"=0 "GateHigh"=1 "GateLow"=2
19	Preset Time	string text	Preset Time	The display text is saved/read.
20	Fine Gain	control index	Fine Gain	FineGain=CSTR(index/8192)
21	TEC Enabled	control index	PX4 Programmer's Guide	Always Enabled = 1
22	RESERVED1	control index	PX4.exe Source Example	Always Enabled = 1
23	PWR_8_5SEL	control index	PX4 Programmer's Guide	Always Enabled = 1
24	RESERVED2	control index	PX4.exe Source Example	Always Enabled = 1
25	PWR_9V	control index	PX4 Programmer's Guide	Always Enabled = 1
26	Input Polarity	control index	Input Polarity	0=Positive, 1=Negative
27	Input Offset	control index	Input Offset	To calculate the Input Offset

				display value: i = (index - 2048) / 1000 InputOffset = cstring(i)+"V"
28	Pole Zero	control index	Pole Zero	
29	Aux Out	control index	Auxiliary Output	
30	HV	control index	High Voltage	
31	TEC	control index	Thermoelectric Cooler	
32	Audio	control index	Audible Counter	
33	Preset Count	setting value	Preset Count	
34	SCA1	packed bit array	Single Channel Analyzers	
35	SCA2	packed bit array	Single Channel Analyzers	
36	SCA3	packed bit array	Single Channel Analyzers	
37	SCA4	packed bit array	Single Channel Analyzers	
38	SCA5	packed bit array	Single Channel Analyzers	
39	SCA6	packed bit array	Single Channel Analyzers	
40	SCA7	packed bit array	Single Channel Analyzers	
41	SCA8	packed bit array	Single Channel Analyzers	
42	Acquisition Mode	control index	Acquisition Mode	
43	MCS Timebase	control index	MCS Timebase	

Current version as of manual release date: 10/2/2009

DPPAPI - DPP Digital Pulse Processor API Reference Manual Version: 2.0

Copyright (C) 2000-2009 Amptek, Inc.

END OF DOCUMENT