DP4/PX4/DP5 Digital Pulse Processor API Help

The DP4/PX4/DP5 Digital Pulse Processor API is an Application Programming Interface Library for the Amptek DP4,PX4 and DP5 Digital Pulse Processors.

DPPAPI Function Groups

- _ DPPAPI Introduction
- Configuration Helper Functions
- Configuration Transfer Functions
- Data Acquisition Functions
- DPP Tune Functions
- DPP USB Additional Functions
- DPPAPI Constructor Functions
- DPPAPI Setup Notes
- PX4 Text Functions
- Status Indicator Functions
- USB Manager Functions

DPPAPI Reference Sections

Functions

Structures

Modules

Help file built: 02/24/09

Functions

- ClearDppData
- CloseDppApi
- CloseUSBDevice
- CreateMCAFileDPPSettings
- DisplayStatusBuffer
- Get80MHzMode
- GetCoarseGainSpinFromText
- GetCoarseGainTextFromSpin
- GetConfigFromBuffer
- GetConfigFromDpp
- GetConfigFromFile
- GetConfigString
- GetDppData
- GetOutputOffsetSpinFromText
- GetOutputOffsetTextFromSpin
- GetPresetTimeStringFromVal
- GetPresetTimeValFromString
- GetPX4Text
- GetSlowThreshSpinFromChannel
- GetSlowThreshSpinFromPercent
- GetSlowThreshStringsFromSpin
- GetStatusBuffer
- GetStatusString
- GetStatusStringFromBuffer
- GetStatusStruct
- GetStatusStructFromBuffer
- GetTempConfigSettings
- GetTempConfigString
- GetTimeToPeakDecimationFromSpin
- GetTimeToPeakSpinFromValues
- GetUSBDppDeviceInfo
- MonitorUSBDppDevices
- OpenDppApi
- OpenUSBDevice
- OpenUSBDeviceEx
- PauseDppData
- ProcessDppData
- ProcessStatusBuffer
- SaveConfigToFile
- SendConfigToBuffer
- SendConfigToDpp
- SendDppVendorRequest
- SetFPGAClockDefault
- SetPX4Text
- SetTempConfigSettings
- TuneFastThreshold
- TuneInputOffset
- TuneInputOffsetInit

Modules

- _DPPAPI Introduction
- Configuration Helper Functions
- Configuration Transfer Functions
- Data Acquisition Functions
- DPP Tune Functions
- DPP USB Additional Functions
- DPPAPI Constructor Functions
- DPPAPI Setup Notes
- PX4 Text Functions
- Status Indicator Functions
- USB Manager Functions

Structures

- DPP_BOOT_STATUS
- DPP_CONFIG_SETTINGS
- DPP_STATUS

_ DPPAPI Introduction

Filename: DppApi.h

Description

Interface to the DPP Application functions.

The DP4/PX4/DP5 Digital Pulse Processor API (DPPAPI) is the Application Programming Interface Library for Amptek DP4, PX4 and DP5 Digital Pulse Processors (DPPs).

Developer Notes

Application programs invoke DPPAPI functions to establish communication with DPP, control its operation, and retrieve data. DPPAPI is distributed in two forms, as an object link library (.LIB) or Dynamic Link Library (.DLL). The application executable programs (.EXE) are created by linking the application code with one of these libraries statically or dynamically (see Microsoft documentation about program linking). The applications can be developed in C, Visual Basic or many other languages.

Version: v20050729_1423

_ DPPAPI Introduction

- DPP_BOOT_STATUS
- DPP_CONFIG_SETTINGS
- DPP_STATUS

Configuration Helper Functions

Filename: DppApi.h

Description

help in the display, control and conversion of DPP configuration data.

All of the configuration settings have underlying support classes. Configuration Helper Functions tap some of the functionality of the underlying classes. The goal is to reduce the amount of support coding necessary to a minimum.

Configuration Helper Functions

- GetCoarseGainSpinFromText
- GetCoarseGainTextFromSpin
- GetOutputOffsetSpinFromText
- GetOutputOffsetTextFromSpin
- GetPresetTimeStringFromVal
- GetPresetTimeValFromString
- GetSlowThreshSpinFromChannel
- GetSlowThreshSpinFromPercent
- GetSlowThreshStringsFromSpin
- GetTimeToPeakDecimationFromSpin
- GetTimeToPeakSpinFromValues

Configuration Transfer Functions

Filename: DppApi.h

Description

Configuration Transfer Functions provide device configuration services.

Configuration Transfer Functions sets and transfers configurations to and from the current DPP, the DPPAPI and selected Amptek DPP configuration files.

Developer Notes

The DPPAPI and the current DPP must be configured before data acquisition operations can begin.

Configuration Transfer Functions

- Get80MHzMode
- GetConfigFromBuffer
- GetConfigFromDpp
- GetConfigFromFile
- GetConfigString
- GetTempConfigSettings
- GetTempConfigString
- SaveConfigToFile
- SendConfigToBuffer
- SendConfigToDpp
- SetFPGAClockDefault
- SetTempConfigSettings

Data Acquisition Functions

Filename: DppApi.h

Description

Control data acquisition.

Spectrum information is read as a block of data and is converted into histogram ready display values.

Data Acquisition Functions

- ClearDppData
- GetDppData
- PauseDppData
- ProcessDppData

DPP Tune Functions

Filename: DppApi.h

Description

Tunes the fast threshold (DP4/PX4/DP5) and input offset (PX4/DP5 only).

PX4 with FPGA/Firmware greater than Version 3.13 or DP4 with FPGA/Firmware greater than Version 4.00 required.

Developer Notes

The source must be removed to auto tune the fast threshold. Status flags are provided to indicate a tuning function successful completion. (See DPP_STATUS.)

DPP Tune Functions

- TuneFastThreshold
- TuneInputOffset
- TuneInputOffsetInit

DPP USB Additional Functions

Filename: DppApi.h

Description

Send Dpp USB Vendor Request (DP4/PX4/DP5).

Developer Notes

Additonal functions are available through Dpp USB Vendor Requests. This is an advanced function and is not supported. See DP4/PX4/DP5 user manuals for Vendor Request function details.

DPP USB Additional Functions

- SendDppVendorRequest

DPPAPI Constructor Functions

Filename: DppApi.h

Description

Opens the DPP Application functions.

DPPAPI Constructor Functions create, initialize and close the DPPAPI. Opening the DPPAPI with OpenDppApi creates and initializes an instance of the DPPAPI. The DPPAPI instance manages and stores DPP data, configurations, and controls communication.

Developer Notes

All other DPPAPI functions need the DPPAPI to be opened before being called. When all DPPAPI operations are completed and before the application is closed, the DPPAPI must be closed with CloseDppApi.

DPPAPI Constructor Functions

- CloseDppApi
- OpenDppApi

DPPAPI Setup Notes

Filename: DppApi.h

Description

Additional setup notes.

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.

DPPAPI Setup Notes

PX4 Text Functions

Filename: DppApi.h

Description

Sets and reads text stored in the PX4.

PX4 Text Functions works for PX4 with FPGA/Firmware Version 3.13 and above.

PX4 Text Functions

- GetPX4Text
- SetPX4Text

Status Indicator Functions

Filename: DppApi.h

Description

Status Indicator Functions monitor and display DPP device information.

Status information is read as a block of data and is converted into display values. Status information includes device identification, spectrum accumulation status, monitors, and boot status.

Status Indicator Functions

- CreateMCAFileDPPSettings
- DisplayStatusBuffer
- GetStatusBuffer
- GetStatusString
- GetStatusStringFromBuffer
- GetStatusStruct
- GetStatusStructFromBuffer
- ProcessStatusBuffer

USB Manager Functions

Filename: DppApi.h

Description

USB Manager Functions provide USB device control.

USB Manager Functions provide USB device control. All USB communications to the DPP require that the USB device be open for the corresponding DPP. When a USB device is opened with OpenUSBDevice communications are opened to the DPP. All USB functions are handled within the DPPAPI.

Developer Notes

Once all USB operations are completed and before the DPPAPI is closed the USB device should be closed with CloseUSBDevice. The number of USB device connections can be monitored with MonitorUSBDppDevices. If a device has been disconnected, MonitorUSBDppDevices will indicate the number of devices has decreased and appropriate action can be taken.

USB Manager Functions

- CloseUSBDevice
- GetUSBDppDeviceInfo
- MonitorUSBDppDevices
- OpenUSBDevice
- OpenUSBDeviceEx

ClearDppData

void __stdcall ClearDppData(void * *objptr*, **BOOLEAN** *isBufferA***)**

Clears the USB DPP spectrum data.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

isBufferA
: Spectrum data buffer selector.

CloseDppApi

void __stdcall CloseDppApi(void * ***objptr*****)**

Closes the current instance of the DPPAPI.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

CloseUSBDevice

void __stdcall CloseUSBDevice(void * *objptr***)**

Closes the USB current device for communications.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

CreateMCAFileDPPSettings

void __stdcall CreateMCAFileDPPSettings(void * *objptr*, LPSTR *pszSettings*, LONG *cSize*)

Creates a MCA acquisition file string that includes all configuration and status settings.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

pszSettings
: DPP devices settings.

cSize
: Character buffer size.

Developer Notes

This function formats the additional information for MCA spectrum file storage. This settings string is appended to the end of the DPP MCA spectrum file after the spectrum data section. This information is used by the Amptek acquisition software.

DisplayStatusBuffer

void __stdcall DisplayStatusBuffer(void * *objptr*, **UCHAR** *Status[]*, **LPSTR** *pszStatus*, **LONG** *cSize***)**

Processes raw status data, stores the status data in the DPPAPI, and returns a status display string.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Status[]
: Status buffer.

pszStatus
: Status character string.

cSize
: Character buffer size.

Developer Notes

The raw status data is read from the DPP by calling GetStatusBuffer. DisplayStatusBuffer is used to update the DPPAPI to the current DPP status and request status display information.

Get80MHzMode

long __stdcall Get80MHzMode(void * ***objptr*****)**

Gets FPGA 80MHz Clock Mode for DP5 only, returns true if is a DP5 and in 80MHz Mode.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Developer Notes

The FPGA 80MHz Clock Mode indicator is set when a DPP status command is issued and a DPP device is attached or when SetFPGAClockDefault sets this value. The DP5 FPGA 80MHz Clock Mode must be set to correctly interpret DP5 configuration file settings.

GetCoarseGainSpinFromText

void __stdcall GetCoarseGainSpinFromText(void * *objptr***, byte *** *Spin***, LPCSTR** *pszCoarseGain***)**

Creates an up-down spin control value from a coarse gain display string.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Spin
: Up-down spin control setting.

pszCoarseGain
: Coarse gain value string.

GetCoarseGainTextFromSpin

void __stdcall GetCoarseGainTextFromSpin(void * *objptr*, **byte** *Spin*, **LPSTR** *pszCoarseGain*, **LONG** *cSize***)**

Creates a coarse gain display string from a selected up-down spin control value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Spin
: Up-down spin control setting.

pszCoarseGain
: Coarse gain value string.

cSize
: Character buffer size.

GetConfigFromBuffer

void __stdcall GetConfigFromBuffer(void * *objptr***)**

Loads the stored configuration read from the DPP (stored in a byte block) into the DPPAPI.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI. configuration buffer read from the dpp stored in a byte block the current device type (default is 1=DP4)

GetConfigFromDpp

void __stdcall GetConfigFromDpp(void * *objptr***)**

Loads the stored configuration from the DPP USB into the DPPAPI.

Defined in: DppApi.h

Parameters

objptr
    Pointer to the DPPAPI.

GetConfigFromFile

void __stdcall GetConfigFromFile(void * *objptr*, **LPCSTR** *pszFilename*, **LONG** *cSize*, **byte** *DPPDeviceType***)**

Loads a configuration from a DPP configuration file into the DPPAPI.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

pszFilename
: Dpp configuration filename.

cSize
: Character buffer size.

DPPDeviceType
: DPP Device Type

Developer Notes

DPP Device Type (1=DP4,2=PX4,3=DP5(DP4 Emulation),4=DP5(Px4 Emulation)) If no device type or an invalid device type is selected, DPPDeviceType defaults to PX4.

GetConfigString

void __stdcall GetConfigString(void * *objptr*, **LPSTR** *pszConfig*, **LONG** *cSize***)**

Creates display formatted copy of the current DPPAPI configuration.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

pszConfig
: Configuration display string.

cSize
: Character buffer size.

GetDppData

int __stdcall GetDppData(void * *objptr*, **long** *DataBuffer[]***)**

Get the USB DPP spectrum data.

Defined in: DppApi.h

Return Value

Returns the number of spectrum channels.

Parameters

objptr
    Pointer to the DPPAPI.

DataBuffer[]
    Holds the DPP spectrum data.

GetOutputOffsetSpinFromText

void __stdcall GetOutputOffsetSpinFromText(void * *objptr***, byte *** *Spin***, LPCSTR** *pszOffset***)**

Creates an up-down spin control value from an output offset display string.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Spin
: Up-down spin control setting.

pszOffset
: Output offset value string.

GetOutputOffsetTextFromSpin

void __stdcall GetOutputOffsetTextFromSpin(void * *objptr*, **byte** *Spin*, **LPSTR** *pszOffset*, **LONG** *cSize***)**

Creates an output offset display string from a selected up-down spin control value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Spin
: Up-down spin control setting.

pszOffset
: Output offset value string.

cSize
: Character buffer size.

GetPresetTimeStringFromVal

void __stdcall GetPresetTimeStringFromVal(void * *objptr*, **long** *lPresetTime*, **LPSTR** *pszPresetTime*, **LONG** *cSize***)**

Creates a preset time display string from a preset time value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

lPresetTime
: Preset time in tenths of a second.

pszPresetTime
: Preset time display string.

cSize
: Character buffer size.

GetPresetTimeValFromString

void __stdcall GetPresetTimeValFromString(void * *objptr***, long *** *lPresetTime***, LPCSTR** *pszPresetTime***)**

Calculates a preset time value from a preset time display string.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

lPresetTime
: Preset time in tenths of a second.

pszPresetTime
: Preset time display string.

GetPX4Text

void __stdcall GetPX4Text(void * *objptr***, LPSTR** *pszPX4Text***, LONG** *cSize***)**

Gets the PX4 stored text.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

pszPX4Text
: PX4 text string.

cSize
: Character buffer size.

GetSlowThreshSpinFromChannel

void __stdcall GetSlowThreshSpinFromChannel(void * *objptr*, **double** *dblChannel*, **byte *** *Spin***)**

Calculates the up-down spin control value for a given slow threshold channel value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

dblChannel
: Slow threshold spectrum channel.

Spin
: Up-down spin control setting.

GetSlowThreshSpinFromPercent

void __stdcall GetSlowThreshSpinFromPercent(void * *objptr*, **double** *dblPercent*, **byte *** *Spin***)**

Calculates the up-down spin control value for a given slow threshold percent value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

dblPercent
: Slow threshold percent of full energy scale.

Spin
: Up-down spin control setting.

GetSlowThreshStringsFromSpin

void __stdcall GetSlowThreshStringsFromSpin(void * *objptr*, **byte** *Spin*, **LPSTR** *pszPercent*, **LONG** *cSizePer*, **LPSTR** *pszChannels*, **LONG** *cSizeChan***)**

Creates slow threshold percent and channel display strings from an up-down spin control value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Spin
: Up-down spin control setting.

pszPercent
: Slow threshold percent display string.

cSizePer
: Size of the percent character buffer.

pszChannels
: Slow threshold channel position display string.

cSizeChan
: Size of the channel character buffer.

GetStatusBuffer

void __stdcall GetStatusBuffer(void * *objptr*, **BOOLEAN** *isBufferA*, **UCHAR** *Status[]***)**

Requests the current DPP device and spectrum status raw data.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

isBufferA
: Spectrum data buffer selector.

Status[]
: Status buffer.

Developer Notes

A spectrum buffer (A or B) is selected to determine which group of Spectrum Accumulation Status information to return.

GetStatusString

void __stdcall GetStatusString(void * *objptr*, **BOOLEAN** *isBufferA*, **LPSTR** *pszStatus*, **LONG** *cSize***)**

Gets a copy of the current status and stores the data in a string.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

isBufferA
: Spectrum data buffer selector.

pszStatus
: Status character string.

cSize
: Character buffer size.

Developer Notes

A spectrum buffer (A or B) is selected to determine which group of Spectrum Accumulation Status information to return.

GetStatusStringFromBuffer

void __stdcall GetStatusStringFromBuffer(void * *objptr*, **BOOLEAN** *isBufferA*, **LPSTR** *pszStatus*, **LONG** *cSize*, **UCHAR** *Status[]***)**

Gets a copy of the current status and stores the data in a string.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

isBufferA
: Spectrum data buffer selector.

pszStatus
: Status character string.

cSize
: Character buffer size.

Status[]
: Status buffer.

Developer Notes

A spectrum buffer (A or B) is selected to determine which group of Spectrum Accumulation Status information to return.

GetStatusStruct

void __stdcall GetStatusStruct(void * *objptr*, **BOOLEAN** *isBufferA*, **DPP_STATUS *** *StatusStruct***)**

Returns the current DPP device and spectrum status data in a status structure.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

isBufferA
: Spectrum data buffer selector.

StatusStruct
: Status structure.

Developer Notes

A spectrum buffer (A or B) is selected to determine which group of Spectrum Accumulation Status information to return.

GetStatusStructFromBuffer

void __stdcall GetStatusStructFromBuffer(void * *objptr*, **BOOLEAN** *isBufferA*, **DPP_STATUS *** *StatusStruct*, **UCHAR** *Status[]***)**

Returns the current DPP device and spectrum status data in a status structure.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

isBufferA
: Spectrum data buffer selector.

StatusStruct
: Status structure.

Status[]
: Status buffer.

Developer Notes

A spectrum buffer (A or B) is selected to determine which group of Spectrum Accumulation Status information to return.

GetTempConfigSettings

void __stdcall GetTempConfigSettings(void * *objptr*, **DPP_CONFIG_SETTINGS *** *CfgSet*, **BOOLEAN** *CurrentUpdate***)**

Copies selected configuration to a DPP_CONFIG_SETTINGS structure.

Defined in: DppApi.h

Parameters

objptr
    Pointer to the DPPAPI.

CfgSet
    Configuration settings.

CurrentUpdate
    Current settings update.

Developer Notes

If UpdateCurrent is selected, the configuration is copied from the current DPPAPI configuration settings. If UpdateCurrent is not selected, the last temporary configuration storage is used.

GetTempConfigString

void __stdcall GetTempConfigString(void * *objptr*, **LPSTR** *pszConfig*, **LONG** *cSize***)**

Creates a display formatted copy of the temporary DPPAPI configuration.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

pszConfig
: Configuration display string.

cSize
: Character buffer size.

GetTimeToPeakDecimationFromSpin

void __stdcall GetTimeToPeakDecimationFromSpin(void * *objptr*, **byte *** *TimeToPeak*, **byte *** *Decimation*, **byte** *Spin***)**

Calculates the time-to-peak and decimation values from an up-down spin control value.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

TimeToPeak
: Time-to-peak value.

Decimation
: Decimation value.

Spin
: Up-down spin control setting.

GetTimeToPeakSpinFromValues

void __stdcall GetTimeToPeakSpinFromValues(void * *objptr*, **byte** *TimeToPeak*, **byte** *Decimation*, **byte *** *Spin***)**

Converts a time-to-peak value to an up-down spin control setting.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

TimeToPeak
: Time-to-peak value.

Decimation
: Decimation value.

Spin
: Up-down spin control setting.

Developer Notes

When a time-to-peak value is known it may be necessary to remotely set an up-down spin control. GetTimeToPeakSpinFromValues uses a time-to-peak value a known decimation value to calculate the correct control setting.

GetUSBDppDeviceInfo

int __stdcall GetUSBDppDeviceInfo(void * *objptr*, **long** *Device*, **long *** *lSerialNumber***)**

Get device information (serial number and type) for the selected usb device.

Defined in: DppApi.h

Return Value

Returns the type Amptek USB device detected.

Amptek DPP USB devices

DPPNONE = 0x0	// none
DPPDP4 = 0x1	// DP4
DPPPX4 = 0x2	// PX4
DPPDP4EMUL = 0x3	// DP5 with DP4 Emulation
DPPPX4EMUL = 0x4	// DP5 with PX4 Emulation (Same as DPPDP5)
DPPDP5 = 0x4	// DP5 with PX4 Emulation

Parameters

objptr
: Pointer to the DPPAPI.

Device
: selected USB DPP device

lSerialNumber
: serial number of device

MonitorUSBDppDevices

int __stdcall MonitorUSBDppDevices(void * *objptr***)**

Monitors the USB current device to determine if connected and communicating.

Defined in: DppApi.h

Return Value

Returns the number of Amptek USB devices detected.

Parameters

objptr
    Pointer to the DPPAPI.

OpenDppApi

void * __stdcall OpenDppApi(void)

Creates and initializes an instance of the DPPAPI.

Defined in: DppApi.h

Return Value

Returns an object pointer handle to the DPPAPI.

OpenUSBDevice

int __stdcall OpenUSBDevice(void * *objptr***)**

Opens the USB current device for communications.

Defined in: DppApi.h

Return Value

Returns the number of Amptek USB devices detected.

Parameters

objptr
: Pointer to the DPPAPI.

OpenUSBDeviceEx

int __stdcall OpenUSBDeviceEx(void * *objptr*, **long** *Device***)**

Opens the USB selected device for communications.

Defined in: DppApi.h

Return Value

Returns the number of Amptek USB devices detected.

Parameters

objptr
: Pointer to the DPPAPI.

Device
: selected USB DPP device

PauseDppData

void __stdcall PauseDppData(void * *objptr*, **BOOLEAN** *boolPauseData***)**

Pauses the USB DPP spectrum data.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

boolPauseData
: Pause the spectrum data.

ProcessDppData

int __stdcall ProcessDppData(void * *objptr*, **long** *DataBuffer[]*, **UCHAR** *RawData[]*, **UINT** *ByteCount***)**

Process the USB DPP spectrum data.

Defined in: DppApi.h

Return Value

Returns the number of spectrum channels.

Parameters

objptr
    Pointer to the DPPAPI.

DataBuffer[]
    Holds the DPP spectrum data.

RawData[]
    Holds pre-processed spectra data.

ByteCount
    Holds pre-processed spectra data byte count.

ProcessStatusBuffer

void __stdcall ProcessStatusBuffer(void * *objptr*, **UCHAR** *Status[]***)**

Processes raw status data and stores the status data in the DPPAPI.

Defined in: DppApi.h

Parameters

objptr
    Pointer to the DPPAPI.

Status[]
    Status buffer.

Developer Notes

The raw status data is read from the DPP by calling GetStatusBuffer. ProcessStatusBuffer is used to update the DPPAPI to the current DPP status without requesting status display information.

SaveConfigToFile

void __stdcall SaveConfigToFile(void * *objptr*, **LPCSTR** *pszFilename*, **LONG** *cSize***)**

Sends the current configuration from the DPPAPI to a DPP configuration file.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

pszFilename
: Dpp configuration filename.

cSize
: Character buffer size.

SendConfigToBuffer

void __stdcall SendConfigToBuffer(void * *objptr***)**

Sends the current configuration from the DPPAPI to a configuration buffer.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI. dppapi configuration stored in a byte block preset setting

SendConfigToDpp

void __stdcall SendConfigToDpp(void * *objptr***)**

Sends the current configuration from the DPPAPI to the DPP USB.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

SendDppVendorRequest

void __stdcall SendDppVendorRequest(void * *objptr*, **int** *Request*, **int** *Value***)**

Sends a generic DPP Vendor Request to the default usb device.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Request
: Vendor request.

Value
: Vendor request value.

Developer Notes

The function SendDppVendorRequest is determined by the Request parameter. See DP4/PX4/DP5 user manuals for Vendor Request function details.

Boot Option Vendor Requests:

BootOptSetBootFlags = 0x92 //Set DP5 boot options

BootOptSetPC5HV = 0x93 //Set PC5 HV

BootOptPC5TECTemp = 0x94 //Set PC5 TEC temperature

BootOptDP5InputOffset = 0x95 //Set DP5 input offset

BootOptSetDP5uCtrADC = 0x96 //Set DP5 microcontroller ADC/Temperature calibration (RESERVED)

BootOptSetDP5SpectrumOffset = 0x97 //Set DP5 spectrum offset

SetFPGAClockDefault

void __stdcall SetFPGAClockDefault(void * *objptr*, **BOOLEAN** *b80MHzMode*, **byte** *DPPDeviceType***)**

Sets FPGA 80MHz Clock Mode indicator for DP5 only

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

b80MHzMode
: 80MHz Mode indicator.

DPPDeviceType
: DPP Device Type.

Developer Notes

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.

SetPX4Text

void __stdcall SetPX4Text(void * *objptr*, **LPCSTR** *pszPX4Text***)**

Sets the PX4 stored text.

Defined in: DppApi.h

Parameters

objptr
    Pointer to the DPPAPI.

pszPX4Text
    PX4 text string.

SetTempConfigSettings

void __stdcall SetTempConfigSettings(void * *objptr*, **DPP_CONFIG_SETTINGS *** *CfgSet*, **BOOLEAN** *CurrentUpdate***)**

Copies the selected settings from a DPP_CONFIG_SETTINGS data structure.

Defined in: DppApi.h

Parameters

objptr
Pointer to the DPPAPI.

CfgSet
Configuration settings.

CurrentUpdate
Current settings update.

Developer Notes

If UpdateCurrent is selected, the temporary configuration is also copied to the current DPPAPI configuration settings. Current DPPAPI configuration settings can be used to configure the DPP or be saved to a configuration file for storage.

TuneFastThreshold

void __stdcall TuneFastThreshold(void * *objptr*)

Tunes the fast threshold.

Defined in: DppApi.h

Parameters

objptr
: Pointer to the DPPAPI.

Developer Notes

The source must be removed before tuning the fast threshold.

The status flag SetFastThreshDone is set (=TRUE) if the fast threshold tuning was successful. If the fast threshold tuning was successful, the configuration must be read back using GetConfigFromDpp to update the DppApi copy of the configuration.

TuneInputOffset

void __stdcall TuneInputOffset(void * ***objptr*****)**

Tunes the input offset. (PX4/DP5 only.)

Defined in: DppApi.h

Parameters

objptr
    Pointer to the DPPAPI.

Developer Notes

Input Offset MUST be properly set before running TuneInputOffset. The initial input offset setting is based on the current input polarity and polezero configuration settings. The TuneInputOffsetInit function checks these values and sets the initial input offset value.

The status flag SetInputOffsetDone is set (=TRUE) if the input offset tuning was successful. If the input offset tuning was successful, the configuration must be read back using GetConfigFromDpp to update the DppApi copy of the configuration.

(See the DppApi Reference and the PX4/DP5 Hardware Description for details.)

Possible input offset tune start settings:

SET TO -2.048V (= 0000) IF THE Input Polarity is NEGATIVE(-) and PoleZero > 0

SET TO -0.100V (= 1948) IF THE Input Polarity is NEGATIVE(-) and PoleZero = 0

SET TO +0.100V (= 2148) IF THE Input Polarity is POSITIVE(+) and PoleZero = 0

SET TO +2.042V (= 4090) IF THE Input Polarity is POSITIVE(+) and PoleZero > 0

TuneInputOffsetInit

void __stdcall TuneInputOffsetInit(void * *objptr***)**

Sets initial values for PX4 input offset tuning. (PX4 only.)

Defined in: DppApi.h

Parameters

objptr
Pointer to the DPPAPI.

DPP_BOOT_STATUS structure

```
struct {
      USHORT BootFlags;
      USHORT HVDACSet;
      USHORT TECDACSet;
      USHORT InputOffsetTuning;
      USHORT uCTempCal;
      USHORT SpectrumOffset;
} DPP_BOOT_STATUS;
```

DPP Boot Options Status Storage Type

Defined in: DppApi.h

Members

BootFlags
: Boot flags (MSB currently unused)

HVDACSet
: HV DAC setting (not used in PX4 mode)

TECDACSet
: TEC DAC setting (not used in PX4 mode)

InputOffsetTuning
: Input offset DAC setting (not used in PX4 mode)

uCTempCal
: uC temp cal

SpectrumOffset
: Spectrum offset

DPP_CONFIG_SETTINGS structure

```
struct {
      byte AcqMode;
      byte MCSTimebase;
      byte MCAChannels;
      byte BufferSelect;
      byte TTLGate;
      byte SlowThreshold;
      long PresetTime;
      long PresetCount;
      long SCA[8];
      byte TimeToPeak;
      byte Decimation;
      byte FlatTop;
      byte PUREnable;
      byte FastThreshold;
      byte RTDSlow;
      byte RTDOn;
      byte RTDFast;
      byte BLR;
      byte BaselineOn;
      byte CoarseGain;
      long FineGain;
      long PoleZero;
      long InputOffset;
      byte InputPolarity;
      byte DetReset;
      long TEC;
      byte HVEnabled;
      long HV;
      byte PreampPower;
      byte AnalogOut;
      byte OutputOffset;
      byte AuxOut;
      byte AudibleCounter;
} DPP_CONFIG_SETTINGS;
```

DPP Configuration Settings Storage Type

Defined in: DppApi.h

Members

AcqMode
: acquisition mode 0=MCA,1=MCS

MCSTimebase
: MCS timebase value (0-15) see CAcqMode

MCAChannels
: number of channels 4=256,3=512,2=1024,1=2048,0=4096,5=8192

BufferSelect
: Holds Buffer Sel A&B,Buffer Sel Hardware,see DPPBufferSelect

TTLGate
: gate input settings, determines events included/excluded from spectrum, see DPPGate

SlowThreshold
: Slow ch threshold, Events w/amp lower not added to spectrum

PresetTime
var holds preset time, used in usb
PresetCount
preset count in selected channels, ch are set in SCA8
SCA[8]
SCA values, LL, UL, and Enable are stored together
TimeToPeak
TimeToPeak register setting
Decimation
decimation setting for pulse shaping
FlatTop
flatop register setting
PUREnable
pile-up rejection enabled
FastThreshold
Fast Ch Threshold, events w/fch amp below this are rejected
RTDSlow
Risetime Discrimination slow threshold
RTDOn
Turns RTD on, and sets the amplitude and timing thresholds
RTDFast
RTD Time Threshold,Events w/HWHM wider than this are rejected
BLR
Baseline Restoration, see udBLR.Value notes for values
BaselineOn
use autobaseline during detector reset
CoarseGain
stores current coarse gain value
FineGain
stores current fine gain value
PoleZero
pole zero adjust value
InputOffset
input offset
InputPolarity
use InvertEnable during detector reset
DetReset
detector reset lockout period
TEC
TEC temperature setting (displayed in Kelvin)
HVEnabled
high voltage setting enable
HV
high voltage setting value
PreampPower
preamp power select value (5v or 8.5v)
AnalogOut
dac enabled and DAC output type,(stobed peak,shaped pulse,dec inp,fast ch)
OutputOffset
Output DAC offset, -64…+63,(signed)(D7-D1) (-500mV to +492mV)
AuxOut
Aux output type
AudibleCounter

audio volume setting

DPP_STATUS structure

```
struct {
      double FPGA;
      double Firmware;
      double SerialNumber;
      byte StatDevInd;
      byte BootStatus;
      byte PwrBtnConfig;
      byte SwConfigRcvd;
      byte SetFastThreshDone;
      byte SetSlowThreshDone;
      byte SetInputOffsetDone;
      double BoardTemp;
      double HVMonitor;
      double TECMonitor;
      double FastCount;
      double SlowCount;
      double AccumulationTime;
      byte StatMcaEnabled;
      byte MCSDone;
      byte PresetCountExpired;
} DPP_STATUS;
```

DPP Status Storage Type

Defined in: DppApi.h

Members

FPGA
  FPGA
Firmware
  firmware revision
SerialNumber
  unit serial number
StatDevInd
  device indicator from status block (0=dp4,1=px4)
BootStatus
  boot status byte (PwrBtnConfig,SwConfigRcvd,McaEnabled)
PwrBtnConfig
  dpp has loaded config from power button
SwConfigRcvd
  software has sent valid config to unit since startup
SetFastThreshDone
  DPP tune fast thresh has successfully done
SetSlowThreshDone
  DPP tune slow thresh has successfully done (NOT USED)
SetInputOffsetDone
  DPP tune input offset has successfully done
BoardTemp
  board temperature monitor value
HVMonitor

high voltage monitor value

TECMonitor

TEC temperature monitor value (displayed in Kelvin)

FastCount

fast channel count

SlowCount

slow channel count

AccumulationTime

real time duration of present data acquisition interval

StatMcaEnabled

mca enabled status, high during acq, preset time resets flag

MCSDone

MCS done flag 1=finished, 0=not finished

PresetCountExpired

indicates presets counts have been reached