DP5 API DLL Functions

1. USB COMMUNICATIONS

1.1 ConnectToDefaultDPP

Connects to the first FW6 DPP USB device found.

- DLL_IMPORT bool ConnectToDefaultDPP();
  - Returns
    - true on success
    - false otherwise.

1.2 ConnectToDPP

Connects to a FW6 DPP USB device with a given serial number.

- DLL_IMPORT bool ConnectToDPP(long lSerialNumber);
  - lSerialNumber
    - holds the serial number used to select the DPP.
  - Returns
    - true on success
    - false otherwise.

1.3 ConnectToDPPIndex

Connects to a FW6 DPP USB device with a given index from 1 to the number of devices.

- DLL_IMPORT bool ConnectToDPPIndex(int idxDevice);
  - idxIndex
    - holds the index (from 1 to the number of devices) used to select the DPP.
  - Returns
    - true on success
    - false otherwise.

NOTES:
Use **GetDppSerialNumber** to identify indexed DPP device.

1.4 CloseConnection

Closes connection with FW6 DPP USB device.

- DLL_IMPORT void CloseConnection();

1.5 CountDppDevices

Counts DPP devices. Only call before connection to DPP.

- DLL_IMPORT int CountDpp();
  - Returns
    - Number of FW6 DPP USB devices found.

2. ACQUISITION CONTROL FUNCTIONS

2.1 ClearData

Clears MCA data.

- DLL_IMPORT void ClearData();

2.2 EnableMCA

Enables spectrum data acquisition.

- DLL_IMPORT void EnableMCA();

2.3 DisableMCA

Disables spectrum data acquisition.

- DLL_IMPORT void DisableMCA();

2.4 ReqestSpectrumData

Requests spectrum data.

- DLL_IMPORT void ReqestSpectrumData(struct Spec *SpectrumOut,
  struct DP5_DP4_FORMAT_STATUS *m_DP5_StatusOut);
  - SpectrumOut
    - Holds spectrum data.
  - m_DP5_StatusOut
    - Holds spectrum status.

2.5 AcquireSpectrum

Test loop function demonstrates a spectrum data acquisition using PRET.

- DLL_IMPORT void AcquireSpectrum(double dblPreset, double dblDelay, int iRefreshMS);
  - dblPreset
    - The number of preset seconds to wait for the test to finish.
  - dblDelay
    - Additional delay to wait to ensure the test has completed.
  - iRefreshMS
    - The number of milliseconds to wait between spectrum requests.

NOTES:
For preset counts set dblPreset to the number of seconds to timeout the test loop if the preset is not reached.

2.6 ConsoleSpectrum

Displays spectrum data and status to stdout.

- DLL_IMPORT void ConsoleSpectrum(struct Spec *SpectrumOut, char * sDppStatusString);
  - SpectrumOut t
    - Spectrum data from a spectrum request.
  - sDppStatusString
    - Status data formatted for display.

3. STATUS FUNCTIONS

3.1 GetDppStatus

Requests DPP Status.

- DLL_IMPORT void GetDppStatus();

NOTES:
GetDppStatus stores the status internally. **GetDppStatus MUST** be called before any other status function. Further processing is required to read the status. Call **DppStatusToString** for a status display. Call **DppStatusToStruct** to save status to a structure for further processing.

3.2 DppStatusToString

Saves the DPP status to a string for display.

- DLL_IMPORT void DppStatusToString(char * sDppStatusString);
  - sDppStatusString
    - Holds a formatted status display string.

NOTES:
Call **GetDppStatus** first. The DPP status is a C char display string.

3.3 DppStatusToStruct

Saves the DPP status to a structure for further testing.

- DLL_IMPORT void DppStatusToStruct(struct DP5_DP4_FORMAT_STATUS *m_DP5_StatusOut);
  - m_DP5_StatusOut
    - Holds DPP status as a structure for further testing..

NOTES:
Call **GetDppStatus** first.

3.4 GetDppSerialNumber

Requests DPP Serial Number.

- DLL_IMPORT long GetDppSerialNumber();
  - Returns
    - The serial number on success
    - 0 (zero) otherwise.

NOTES:
Use with **ConnectToDPPIndex** to identify indexed DPP device. DPP connection must by open before calling GetDppSerialNumber.

4. CONFIGURATION FUNCTIONS

4.1 ReadDppConfigurationFromHardware

Requests DPP Configuration.

- DLL_IMPORT void ReadDppConfigurationFromHardware();

NOTES:
The hardware configuration is stored internally. **ReadDppConfigurationFromHardware MUST** be called before calling hardware configuration processing functions.

4.2 DppHwConfigToString

Save hardware configuration to string.

- DLL_IMPORT int DppHwConfigToString(bool bAddComments, char * strConfigOut);
  - bAddComments
    - Set to true to append comments.
  - strConfigOut
    - C string buffer holds the hardware configuration.
  - Returns
    - The size of the output string on success
    - 0 (zero) otherwise.

NOTES:
Call **ReadDppConfigurationFromHardware** first.

4.3 PresetModeToString

Saves the preset mode to a string.

- DLL_IMPORT void PresetModeToString(char * strPresetModeOut);
  - strPresetModeOut
    - C string buffer holds the preset mode.

NOTES:
Call **ReadDppConfigurationFromHardware** first.
Displays two lines of Preset mode information.

Preset Mode Indicators:
Acq=Preset Acquisition Time (Except MCA8000D)
Real=Preset Real Time
Live=Preset Live Time (Only MCA8000D)
Cnt=Preset Count

PX5 with 3 Preset Modes Set:
Preset Mode: Cnt/Acq/Real
Preset Settings: 10000/20.0/25.000

PX5 without Preset Modes Set:
Preset Mode: None
Preset Settings:

4.4 SendCommandString

Sends an ASCII command string to the DPP device.

- DLL_IMPORT bool SendCommandString(char strCMD[]);
  - strCMD
    - String containing ASCII commands.
  - Returns
    - true on success
    - false otherwise.

4.5 SendConfigFileToDpp

Sends a DPP configuration file to the DPP device.

- DLL_IMPORT bool SendConfigFileToDpp(char strFilename[]);
  - strFilename
    - String containing the configuration filename.
  - Returns
    - true on success
    - false otherwise.

4.6 SendScaToDpp

Sends a DPP configuration file with SCA settings to the DPP device.

- DLL_IMPORT bool SendScaToDpp(char strFilename[]);
  - strFilename
    - String containing the SCA configuration filename.
  - Returns
    - true on success
    - false otherwise.

4.7 ShortenCfgCmds

Tests and shortens ASCII command strings to be sent to hardware.

- DLL_IMPORT int ShortenCfgCmds(char strCfgIn[], char * strCfgOut);
  - strCfgIn
    - String to be shortened.
  - strCfgOut
    - Shortened string.
  - Returns
    - The size of the output string on success
    - 0 (zero) otherwise.

NOTES:
If an ASCII command is only marginally larger than the 512 byte limit, this function may be used to shorten the string enough to be sent without being split. ASCII command strings must be 512 bytes or less. If the ASCII command string is greater than 512 bytes, the command string must be split into 512 byte or smaller stings and sent separately.

4.8 DisplayPresets

Sends Presets to stdout.

- DLL_IMPORT void DisplayPresets();

4.9 ReadConfigFile

Reads a configuration file and returns a processed configuration that can be sent to a DPP device.

- DLL_IMPORT int ReadConfigFile(char strFilename[], char * strCfgOut, char * strSplitCfgOut);
  - strFilename
    - String containing the configuration filename
  - strCfgOut
    - Holds ASCII configuration string.
  - strSplitCfgOut
    - Holds ASCII configuration string remainder if too large for one packet.
  - Returns
    - The total size of the ASCII configuration string in bytes.

NOTES:
If an ASCII command is larger than the 512 byte limit, the remainder of the ASCII is in strCfgOut.
ReadConfigFile produces ASCII commands that can be sent with SendCommandString.
ReadConfigFile removes commands not used by the current device and shortens or splits commands when necessary.

5. SPECTRUM FILE FUNCTIONS

5.1 GetStartTime

Saves the start time into a display string and a Date Time Stamp.

- DLL_IMPORT int GetStartTime(char StartTime[], char strDTS[]);
  - StartTime
    - Start time display string. Used for MCA file.
  - strDTS
    - Start time DTS. Used for MCA file.

5.2 SaveMCADataToFile

Saves spectrum data to Amptek spectrum (.mca) file.

- DLL_IMPORT void SaveMCADataToFile(char * strFilename, struct Spec *SPECTRUM, struct SpecFile *sfInfo);
  - strFilename
    - Filename of MCA file to be saved.
  - SPECTRUM
    - Spectrum plot data.
  - sfInfo
    - Status and other acquisition information.

6. DIAGNOSTICS

6.1 SetDisplayCfg

Enables configuration display to stdout.

- DLL_IMPORT void SetDisplayCfg(bool bShowConsoleCfg);
  - bShowConsoleCfg
    - Set to true to enable.

6.2 SetDisplayDebugInfo

Display general debug information to stdout.

- DLL_IMPORT void SetDisplayDebugInfo(bool bShowConsoleDebug);
  - bShowConsoleDebug
    - Set to true to enable.

6.3 SetDisplaySpectrum

Display a text character spectrum to stdout.

- DLL_IMPORT void SetDisplaySpectrum(bool bShowConsoleSpectrum);
  - bShowConsoleSpectrum
    - Set to true to enable.

7. USING LIBUSB USB COMMUNICATIONS

7.1 Identifying and Connecting to DPP Devices

To count the number of devices attached to a system, call CountDppDevices before attaching to a device. Each DPP device is assigned an index from 1 to the number of devices. This index is for identifying and selecting devices using usb. If only one device will be attached, ConnectToDefaultDPP is the best choice to connect to the DPP to begin communications. If many devices will be attached ConnectToDPPIndex or ConnectToDPP can be used.

ConnectToDPPIndex works best for cases where you may not know the serial number of the devices you are connecting to. Once connected to the DPP device you can request the serial number to identify the specific device communicating. ConnectToDPPIndex is run in a loop from 1 to the number of devices found. When the first available device is found the loop exits with the first available device connected. If you had three devices connected, the first instance of the application will connect to device 1, the second to device 2, and so on.

ConnectToDPP works when each instance of an application is assigned a DPP device. ConnectToDPP requires the DPP device serial number to connect.

7.2 Running DPP Devices in Parallel

One or more DPP devices can be attached to a system at the same time.
The API can only communicate with one DPP device at a time.
To communicate with multiple DPP devices at the same time multiple applications are run (easy) or a multithreaded application can call multiple instances of the API in the same application (advanced).