Application Program Interface (API)
for CPS Model 3601-3605 Power Supplies

Introduction

CPS provides a free software development kit (SDK) for CPS model 3600 series of power supplies. The SDK includes an API that allows application software to communicate with model 3601 through 3605 devices. The API is compatible with all variants of model 360x and supports multiple devices connected to a single computer. This document describes API functions and operation, and is intended for software developers who create application programs for model 360x series devices.

Windows SDK

The Windows SDK includes the following files:

- `c3601.dll` – API executable. This must reside in the application program's directory or in the system's DLL search path. Note that this is a 32-bit DLL and a 64-bit DLL for release and debug.
- `c3601.lib` – Linker library for the API executable. This is used by the object file linker when building application programs.
- `c3601win.h` – Header file containing API constants and function declarations. This must be included in C/C+ source code files that call API functions.
- `c3601.h` – Header file containing device constants, error codes , and error flag declarations. This must be included in C/C+ source code files that call API functions.
- A demo program, `360x_ctrl.exe`, and its source files: `3601test.c`, `3601test.h`, `3601test.rc`, `c3601.h, c3601win.c, and c3601win.h`, which shows how to call API functions.

Port number, Dev number

Most of the API functions use a serial port argument (`port`) to specify a particular power supply. Each device must be assigned a unique `port` value in the range 0 to 7. Sub-devices, within the power supply, are identified by the `dev` code. Valid values for the `dev` code are `DEV_A, DEV_E, and DEV_F`. The definition for these can be found in the c3601.h header file.

Error codes

Unless otherwise noted, the value returned by an API function is an error code. Zero is returned if a function executes without error, otherwise a non-zero value is returned. When a non-zero value is returned, it may be one of the error codes listed in c3601.h or it may be a system error value specific to the operating system.

API functions

C3601_OpenPort

```
int C3601_OpenPort(int port, int baudrate);
```

This function must be called once for each power supply to enable subsequent communication with the device. No communications are possible with a device until it has been opened by this function.

`port` specifies the serial COM port to which the device is connected. In Windows systems, the value of this argument equals the COM port number minus one. For example, COM1=0, COM2=1, etc.

`baudrate` value for 360x series devices is 57600 baud.

Example:

```
C2600_OpenDevice(0, 57600); // open device on COM1
```

C3601_ClosePort

```
int C3601_ClosePort(int port);
```

This function closes a device; it must be called once for each open device before closing the application program. After calling this, further communication with the device is prohibited.

C3601_SetValue

```
int C3601_SetValue(int port, int dev, double *v_or_i, int enab);
```

This function programs the setpoint for the Accelerator, Extractor, or Filament. The `v_or_i` parameter, represents a desired output voltage when `dev = DEV_A or DEV_E,` or it represents the desired current output when `dev = DEV_F. The enab` parameter will enable the output when 1, else disable the output when 0.

- `DEV_A indicates the main accelerator voltage`
- `DEV_E indicates the secondary lens or bias voltage`
- `DEV_F indicates the filament current output`

ReadMeter's

```
double C3601_GetVoltage(int port, int dev, int *err);
double C3601_GetCurrent(int port, int dev, int *err);
double C3601_GetTempearture(int port, int dev, int *err);
```

These functions measures one of the internal device meters and returns the meter value. The `dev` argument specifies the device to read. (see above), and also determines the units of the value returned.

The `err` parameter define an optional point to an integer location for storing error values, if any. Set err to 0 to disable the return of an error message.

Function	Return Units
`C3601_GetVoltage`	Volts
`C3601_GetCurrent`	Microamps
`C3601_GetTempearture`	Degrees C

C3601_GetVersion

```
int C3601_GetVersion(int port, int dev);
```

This function returns the version number for the given device on the given port.

C3601_GetStatusFlags

```
int C3601_GetStatusFlags(int port);
```

This function reads status flags from a device. See the header file for a list of status flags.

C3601_GetFaultFlags

```
int C3601_GetFaultFlags(int port, int dev);
```

This function reads fault flags from a device and then clears the `C3601_STATUS_RESET` and `C3601_STATUS_COMERR` flags. The fault flags are returned `by the function`. See the header file for a list of fault flags.

C3601_SetWD

```
int C2600_SetWD(int port, int dev, unsigned int msec);
```

This function configures a device's watchdog timer. The `msec` argument specifies the timer interval in milliseconds. The maximum allowed value is 2550, which corresponds to a 2.55 second interval. Specify `msec=0` to disable the watchdog timer.

C3601_GetAlarm

```
int C3601_GetAlarm(int port);
```

This function reads the status of the Alarm line.

C3601_Reset

```
int C3601_Reset(int port, int dev, int type);
```

This function sets the power supply output voltage to 0 V and clears all communication buffers. Set `type=0` for soft reset, or `type=1` for hard reset.

C3601_GetErrorString

```
int C3601_GetErrorString(char *buf, int maxlen);
```

This function maps an API error code to an associated descriptive string. A maximum of `maxlen` characters is placed in memory pointed to by `buf`.

Example

```
char buf[80];
int errcode = C3601_OpenPort(0, 0);
if (errcode != 0) {
   C3601_GetErrorString(buf, 80)
   printf(" C3601_OpenPort failed: %s\n", buf);
}
```