Promira Serial Platform System User Manual

1 Revision History
- 1.1 Changes in version 1.35
- 1.2 Changes in version 1.34
- 1.3 Changes in version 1.33
- 1.4 Changes in version 1.32
- 1.5 Changes in version 1.31
- 1.6 Changes in version 1.30

2 General Overview

3 Applications

4 Hardware Specifications
- 4.1 Pinouts
- 4.2 USB 2.0 Compliance
- 4.3 Physical Specifications
- 4.4 Promira Boot Process

5 Software
- 5.1 Compatibility
- 5.2 Connectivity
- 5.3 Detecting IP addresses
- 5.4 Dynamically Linked Library

6 Firmware
- 6.1 Field Upgrades

7 API Documentation
- 7.1 Introduction
- 7.2 General Data Types
- 7.3 Notes on Status Codes
- 7.4 Application Management Interface
- 7.5 Error Codes

8 Legal / Contact
- 8.1 Disclaimer
- 8.2 Life Support Equipment Policy
- 8.3 Contact Information

1 Revision History

Top

1.1 Changes in version 1.35

Added auto start feature.

Updated text for upgrades(001)

Top

1.2 Changes in version 1.34

Improved the connection to Windows.

Top

1.3 Changes in version 1.33

Added features for Promira platform with eSPI Analysis Application.

Updated Applications section.

Top

1.4 Changes in version 1.32

Fixed the connectivity issue of the first attempt to connect the Promira platform to the Promira API pm_load function after the device reset.

Top

1.5 Changes in version 1.31

Added features for Promira platform I^2C Active Level 2 Application and SPI Active Level 3 Application.

Top

1.6 Changes in version 1.30

Added features for Promira platform SPI Active Level 2 Application.

2 General Overview

The Promira Serial Platform with the I^2C/SPI Active applications and eSPI Analysis application supports I^2C master/slave active modes; Single, Dual, and Quad SPI master/slave active modes; and Single, Dual and Quad eSPI analysis modes. The Promira platform supports up to 8 SPI SS/CS signals, up to 16 GPIO signals, and up to 11 eSPI Digital IO signals depending on purchased application. The Promira platform connects to an analysis computer via Ethernet or Ethernet over USB. The applications installed on the Promira Serial Platform are field-upgradeable and future-proof.

3 Applications

Table 1 (1) : Applications Description

Application	Max Bitrate (Master/Slave MHz)	Max GPIO Signals	Digital IO Signals	Max SS/CS Signals	Slave Response Level (1)	Slave Response Capability (1)	Slave Response Size (1)
I^2C Active Level 1	1/1	6	NA	NA	Advanced Slave	1 slave address, 1 response	255 Bytes
I^2C Active Level 2	3.4/3.4	12	NA	NA	Advanced Slave	1 slave address, 1 response	255 Bytes
SPI Active Level 1	12.5/8	6	NA	1	Advanced Slave	1 response	256 Bytes
SPI Active Level 2	40/20	12	NA	3	Advanced Slave, Ultra Slave	Advanced Slave: 1 response. Ultra Slave: Multiple responses	256 Bytes
SPI Active Level 3	80/20	16	NA	8	Advanced Slave, Ultra Slave	Advanced Slave: 1 response. Ultra Slave: multiple responses	Advanced Slave: 256 Bytes. Ultra Slave: Up to 64 MBytes

eSPI Analysis	66	NA	11	2	NA	NA	NA

Notes:

(1) Ultra Slave will be released at a future date.

4 Hardware Specifications

Top

4.1 Pinouts

4.1.1 Pin Description

Table 2 (1) : Pin Description - Target Connector

Pin	Symbol	Description
4	V_{TGT}	Software configurable Vcc target power supply. NC/3.3V/5V.
6	V_{TGT}	Software configurable Vcc target power supply. NC/3.3V/5V.
22	V_{IO}	Software configurable Vcc IO level power supply. NC/0.9V to 3.45V.
24	V_{IO}	Software configurable Vcc IO level power supply. NC/0.9V to 3.45V.
2, 10, 12, 16, 18, 28, 30, 34	GND	Ground Connection.

Note:

(1) The pin description in this table is for Promira FW v1.30/v1.30 and above.

Top

4.2 USB 2.0 Compliance

The Promira Serial Platform is USB 2.0 compliant and will operate as a high speed (480 Mbps) device on a USB 2.0 hub or host controller. For additional information see table 15.

Top

4.3 Physical Specifications

- Dimensions: W x D x L: 77.5 mm x 29.2 mm x 115.6 mm (3.05" x 1.15" x 4.55")
- Weight: 153 g (0.34 lbs)

Top

4.4 Promira Boot Process

4.4.1 Promira General Boot Process

1. Promira platform is off, and 3 leds are off
2. Power on Promira platform by connecting it to computer USB connector.
3. In about 1 s right led turns red.
4. In about 5 s right led turns blue.
5. Issue Promira API function pm_load, or connect Promira platform to Control Center Serial or Flash Center.
6. In about 1 us right led turns green, and Promira platform is ready to transfer data.

4.4.2 Promira FW Update Boot Process

1. Right led is green, and Promira platform is ready to transfer data.
2. Issue FW update in Control Center Serial or Promira Update Utility.
3. 3 leds turn off.
4. In about 1 s right led turns red.

5. In about 5 s all 3 leds turn blue.
6. Promira platforms appears as a hard drive on the computer.
7. Copy pmu file to Promira hard drive on the computer
8. Eject safely the Promira hard drive from the computer.
9. Reboot Promira platform by disconnecting and connecting the USB cable.
10. In about 1 s right led turns red.
11. In about 5 s all 3 leds turn blue.
12. In about 5 s all 3 leds flash blue (one at a time).
13. Promira platform automatically reboots: Bullets 1-6 in General Boot Process section are occurred.

4.4.3 Promira License Update Boot Process

The steps are similar to the Promira FW Update Boot Process steps. However, it is faster, and without step 12.

5 Software

Top

5.1 Compatibility

Software compatibility information for the Promira Serial Platform, including supported versions of **Windows, Linux, and macOS**, is provided on the respective software product pages under System Requirements.

- Control Center Serial Software
- Flash Center Software
- Data Center Software
- Promira Software API I2C/SPI Active
- Promira Software API eSPI Analysis
- Promira Aardvark Wrapper Software API

Top

5.2 Connectivity

There are two ways to connect to the Promira Serial Platform: via USB or via Ethernet. No additional device drivers are required for using either method.

5.2.1 USB

The Promira Serial Platform uses Ethernet over USB, which allows the host software to connect to the adapter via an IP address. To use this interface, connect the device to your PC with a USB cable and follow the instructions below to set up the connection on the PC.

For Ethernet over USB, the Promira Serial Platform is a DHCP server that dynamically distributes network configuration parameters, such as IP addresses for interfaces and services.

Windows

Top

1. Connect Promira to PC with USB cable.
2. Promira Serial Platform in **Device Manager** (Please note some Windows versions may display "Total Phase Promira Serial Platform" in this window.):

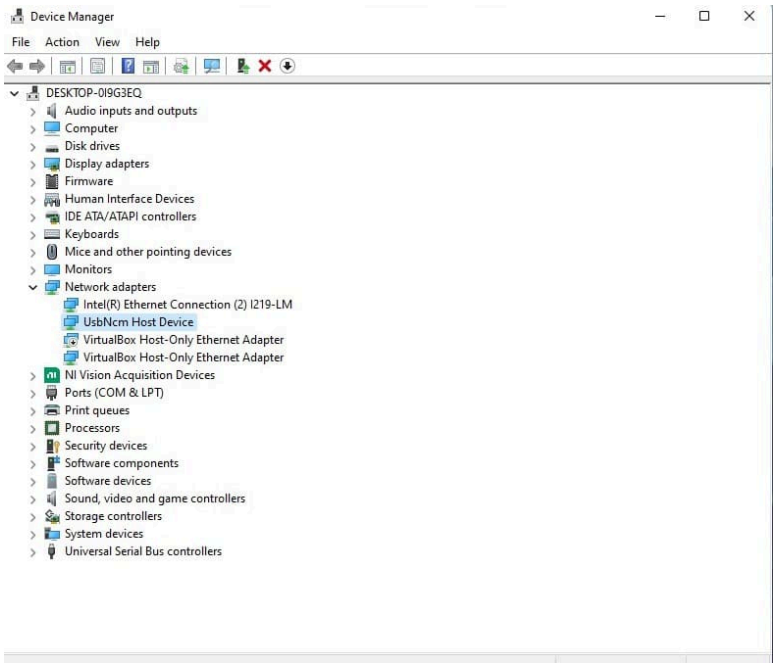


***Figure 1**: Windows Device Manager window.*

3. Promira Serial Platform in **Network Settings**:

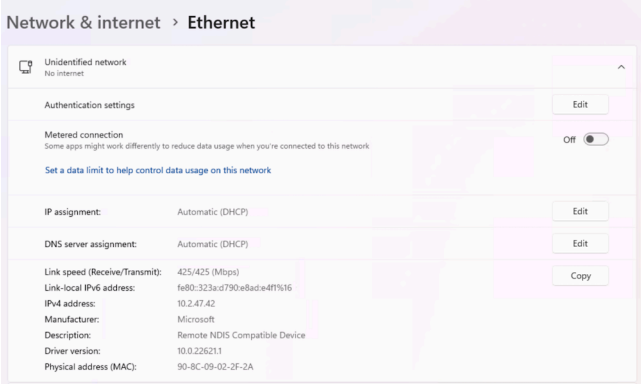


***Figure 2**: Windows Network Settings window.*

4. Verify connectivity with Control Center Serial Software.

Top

Linux

1. Download the Promira Serial Platform Linux support files from the website and follow the instructions in the README.txt file.
2. Connect Promira to PC with USB cable.
3. Use `ifconfig -a` to determine the network interface of Promira. If you do not recognize which one is the new interface, compare the lists from `ifconfig -a` before and after plugging in the device.
4. The Promira Serial Platform will be shown as tpp*x*.

Top

Mac OS X

1. Connect Promira to PC with USB cable.
2. Select **Network** under **System Preferences.**
3. Select **Total Phase Promira Platform.**

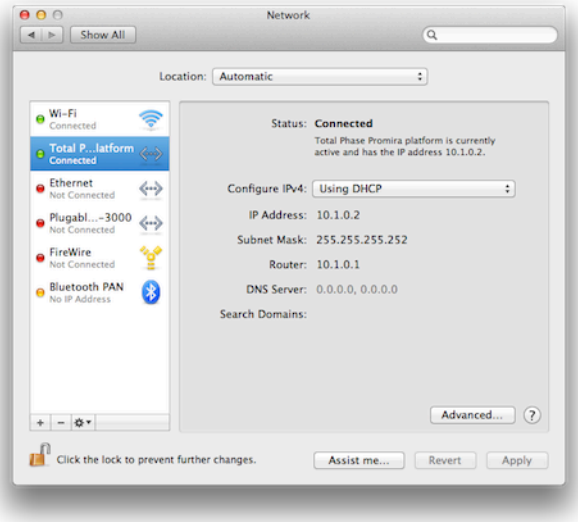


***Figure 3** : Mac OS X Network Preferences window.*

4. Select **Using DHCP** from the **Configure IPv4:** dropdown list box.
5. Select **Apply** to apply the changes.

5.2.2 Ethernet

The Promira platform can be connected to the computer through the Ethernet port. The Promira network preference for Ethernet is configurable by the Control Center Serial. The Promira platform can be configured to static IP addressing or dynamic IP addressing (DHCP). The default network preferences of the Promira platform for Ethernet is a static and its IP address is 192.168.11.1. This IP address can be changed using the **promira** command-line application provided in the **util** folder in the Promira API package. For more details see the Control Center Serial manual section 2.8 Configuring Network Preference and the README.txt file in the API package.

Top

5.3 Detecting IP addresses

To detect the IP addresses to which the Promira Serial Platforms are attached, use the `pm_find_devices` routine as described in following API documentation. Alternatively, the Control Center software can be used to list the available devices.

Top

5.4 Dynamically Linked Library

The Promira requires the Promira DLL to operate and is only compatible with the Promira Serial Platform.

In addition to the Promira DLL, the Aardvark Compatibility DLL is provided to make the Aardvark API available for legacy and compatibility purposes.

5.4.2 DLL Location

Total Phase provides language bindings that can be integrated into any custom application. The default behavior of locating the Promira DLL and the Aardvark Compatibility DLL is dependent on the operating system platform and specific programming language environment. For example, for a C or C++ application, the following rules apply:

On a Windows system, this is as follows:

1. The directory from which the application binary was loaded.
2. The applications current directory.
3. 32-bit system directory (for a 32-bit application). Examples:
   - `c:\Windows\System32` [Windows 7/8 32-bit]
   - `c:\Windows\SysWow64` [Windows 7/8 64-bit]
4. 64-bit system directory (for a 64-bit application). Examples:
   - `C:\Windows\System32` [Windows 7/8 64-bit]

5. The Windows directory. (Ex: `c:\Windows`)
6. The directories listed in the PATH environment variable.

On a Linux system, this is as follows:

1. First, search for the shared object in the application binary path. If the `/proc` filesystem is not present, this step is skipped.
2. Next, search in the applications current working directory.
3. Search the paths explicitly specified in `LD_LIBRARY_PATH`.
4. Finally, check any system library paths as specified in `/etc/ld.so.conf` and cached in `/etc/ld.so.cache`.

On a Mac OS X system, this is as follows:

1. First, search for the shared object in the application binary path.
2. Next, search in the applications current working directory.
3. Search the paths explicitly specified in `DYLD_LIBRARY_PATH`.
4. Finally, check the /usr/lib and /usr/local/lib system library paths.

If the DLL is still not found, an error will be returned by the binding function. The error code is `PM_UNABLE_TO_LOAD_LIBRARY` for the management API and `PS_APP_UNABLE_TO_LOAD_LIBRARY` for the application API.

5.4.3 DLL Versioning

The Promira Compatibility DLL checks to ensure that the firmware of a given device is compatible. Each DLL revision is tagged as being compatible with firmware revisions greater than or equal to a certain version number. Likewise, each firmware version is tagged as being compatible with DLL revisions greater than or equal to a specific version number.

Here is an example.

```
DLL v1.20: compatible with Firmware >= v1.15 Firmware v1.30: compatible
with DLL >= v1.20
```

Hence, the DLL is not compatible with any firmware less than version 1.15 and the firmware is not compatible with any DLL less than version 1.20. In this example, the version number constraints are satisfied and the DLL can safely connect to the target firmware without error. If there is a version mismatch, the API calls to open the device will fail. See the API documentation for further details.

6 Firmware

Top

6.1 Field Upgrades

6.1.1 Upgrade Philosophy

The Promira Serial Platform is designed so that its internal firmware (FW) and License can be upgraded by the user, thereby allowing the inclusion of any additional applications purchased, software support renewals, performance enhancements, or critical fixes available after the receipt of the device.

The Promira Serial Platform shows multiple FW version numbers in the Control Center Serial GUI Configure Adapter window, Flash Center GUI - Add Adapters window, Data Center GUI connect window, and the Promira Utility window. The first one is the version number of the system FW and the others are the version numbers of the various Promira applications. For example, the FW numbers 1.10/1.00/x.xx represent FW system v1.10 and FW application v1.00 and so on.

Promira API has four main FW functions: pm_version, pm_app_version, ps_app_version, and pa_app_version. Promira API function pm_version returns FW system version and Promira API functions pm_app_version, ps_app_version, and pa_app_version return FW application versions.

6.1.2 Upgrade Procedure

Please refer to Section 7.1 of the Control Center Serial Software User Manual for instructions on installing a new license or firmware update.

7 API Documentation

Top

7.1 Introduction

The Promira API documentation that follows is oriented toward the Promira Rosetta C bindings. The set of Promira API functions and their functionality is identical regardless of which Rosetta language binding is utilized. The only differences will be found in the calling convention of the functions. For further information on such differences please refer to the documentation that accompanies each language bindings in the Promira API Software distribution.

Top

7.2 General Data Types

The following definitions are provided for convenience. All Promira data types are unsigned.

```
typedef unsigned char u08; typedef unsigned short u16; typedef unsigned
int u32; typedef unsigned long long u64; typedef signed char s08; typedef
signed short s16; typedef signed int s32; typedef signed long long s64;
typedef float f32;
```

Top

7.3 Notes on Status Codes

Most of the Promira API functions can return a status or error code back to the caller. The complete list of status codes is provided at the end of this chapter and in the Promira applications' user manuals. All of the error codes are assigned values less than 0, separating these responses from any numerical values returned by certain API functions.

Top

7.4 Application Management Interface

7.4.1 Application Management

Top

Find Devices (pm_find_devices)

```
int pm_find_devices (int num_devices, u32 * devices);
```

Get a list of IP addresses to which Promira adapters are attached.

Arguments

num_devices	maximum size of the array
devices	array into which the IP addresses are returned

Return Value

This function returns the number of devices found, regardless of the array size.

Specific Error Codes

None.

Details

Each element of the array is 4 byte integer value represented IP address. For instance, "192.168.1.2" is 0x0201A8C0.

Two IP addresses to same device might be returned when both Ethernet and Ethernet over USB are enabled.

If the input array is NULL, it is not filled with any values.

If there are more devices than the array size (as specified by `num_devices`), only the first `num_devices` IP addresses will be written into the array.

Top

Find Devices (pm_find_devices_ext)

```
int pm_find_devices_ext (int num_devices, u32 * devices, int num_ids, u32
* unique_ids int num_statuses u32 * statuses);
```

Get a list of IP addresses and unique IDs to which Promira Serial Platforms are attached.

Arguments

num_devices	maximum number of IP addresses to return

devices	array into which the IP addresses are returned
num_ids	maximum number of unique IDs to return
unique_ids	array into which the unique IDs are returned
num_statuses	maximum number of statuses to return
statuses	array into which the statuses are returned

Return Value

This function returns the number of devices found, regardless of the array sizes.

Specific Error Codes

None.

Details

This function is the same as `pm_find_devices()` except that is also returns the unique IDs of each Promira adapter. The IDs are guaranteed to be non-zero if valid.

The IDs are the unsigned integer representation of the 10-digit serial numbers.

The number of devices and IDs returned in each of their respective arrays is determined by the minimum of `num_devices`, `num_ids`, and `statuses`.

If status is PM_DEVICE_NOT_FREE, the device is in-use by another host and is not ready for connection.

Open a Promira Serial Platform (pm_open)

Top

```
Promira pm_open (const char * net_addr);
```

Open a connection to a Promira Serial Platform.

Arguments

net_addr	net address of the Promira Serial Platform. It could be an IPv4 address or a host name.

Return Value

This function returns a Promira handle, which is guaranteed to be greater than zero if valid.

Specific Error Codes

PM_UNABLE_TO_OPEN	The specified net address is not connected to a Promira Serial Platform.
PM_INCOMPATIBLE_DEVICE	There is a version mismatch between the DLL and the firmware. The DLL is not of a sufficient version for interoperability with the firmware version or vice versa.

Details

None.

Close the Promira Device (pm_close)

Top

```
int pm_close (Promira promira);
```

Close the connection to the Promira adapter.

Arguments

promira	handle of the connection to the Promira Serial Platform to be closed

Return Value

The number of devices closed is returned on success. This will usually be 1.

Specific Error Codes

None.

Details

If the promira argument is zero, the function will attempt to close all possible handles, thereby closing all connections to Promira Serial Platforms.

Version (pm_version)

Top

```
int pm_version (Promira promira PromiraVersion * version);
```

Return the version matrix for the system connected to the given handle.

Arguments

promira	handle of the connection to the Promira Serial Platform
version	pointer to pre-allocated structure

Return Value

A status code is returned with PM_OK on success.

Specific Error Codes

None.

Details

The PromiraVersion structure describes the various version dependencies of application components. It can be used to determine which component caused an incompatibility error.

```
struct PromiraVersion { /* Software, firmware, and hardware versions. */
u16 software; u16 firmware; u16 hardware; /* FW requires that SW must be
>= this version. */ u16 sw_req_by_fw; /* SW requires that FW must be >=
this version. */ u16 fw_req_by_sw; /* API requires that SW must be >=
this version. */ u16 api_req_by_sw; /* (year << 24) | (month << 16) |
(day << 8) | build. */ u32 build; };
```

If the handle is 0 or invalid, only software, fw_req_by_sw, and api_req_by_sw version are set.

Sleep (pm_sleep_ms)

Top

```
int pm_sleep_ms (u32 milliseconds);
```

Sleep for given amount of time.

Arguments

milliseconds	number of milliseconds to sleep

Return Value

This function returns the number of milliseconds slept.

Specific Error Codes

None.

Details

This function provides a convenient cross-platform function to sleep the current thread using standard operating system functions.

The accuracy of this function depends on the operating system scheduler. This function will return the number of milliseconds that were actually slept.

Status String (pm_status_string)

Top

```
const char *pm_status_string (int status);
```

Return the status string for the given status code.

Arguments

status	status code returned by a Promira application function.

Return Value

This function returns a human readable string that corresponds to status. If the code is not valid, it returns a NULL string.

Specific Error Codes

None.

Details

None.

Applications (pm_apps)

Top

```
int pm_apps (Promira promira, u16 apps_size, u08 * apps);
```

Return the installed applications.

Arguments

promira	handle of the connection to the Promira Serial Platform
apps_size	number of bytes in `apps`
apps	buffer to place the applications string

Return Value

The length of the apps string in bytes.

Specific Error Codes

PM_INVALID_LICENSE	The installed license is corrupt or invalid.

Details

The `apps` argument will be filled with a colon separated string containing the names of applications that are installed for this device.

Pass a value of `0` for `apps` and `apps_size` to request the total length of the applications string. Note this length does not include the null terminating character.

Licensed Applications (pm_licensed_apps)

Top

```
int pm_licensed_apps (Promira promira, u16 apps_size, u08 * apps);
```

Return the licensed applications.

Arguments

promira	handle of the connection to the Promira Serial Platform
apps_size	number of bytes in `apps`
apps	buffer to place the licensed applications string

Return Value

The length of the apps string in bytes.

Specific Error Codes

None.

Details

The `apps` argument will be filled with a colon separated string containing the names of applications that are licensed for this device.

Pass a value of `0` for `apps` and `apps_size` to request the total length of the applications string. Note this length does not include the null terminating character.

Application Version (pm_app_version)

Top

```
int pm_app_version (Promira promira, const char * app_name,
PromiraVersion * version);
```

Return the version matrix for the given application in the system connected to the given handle.

Arguments

promira	handle of the connection to the Promira Serial Platform
app_name	application name
version	pointer to pre-allocated structure

Return Value

A status code is returned with `PM_OK` on success.

Specific Error Codes

None.

Details

This function is to check the version number of the application without loading it.

Only `firmware`, `hardware`, and `sw_req_by_fw` version are set.

Launch an Application (pm_load)

Top

```
int pm_load (Promira promira, const char * app_name);
```

Launch an application.

Arguments

promira	handle of the connection to the Promira Serial Platform
app_name	application name to be launched

Return Value

A Promira status code is returned with `PM_OK` on success.

Specific Error Codes

PM_APP_NOT_FOUND	There is no application with the specified name.
PM_UNABLE_TO_LOAD_APP	Unable to load the application.
PM_APP_ALREADY_LOADED	There is the auto started application. Use `pm_load_ext` to forcibly load a new application.

Details

This function works same as `pm_load_ext` with `PM_LOAD_NO_FLAGS`.

Launch an Application (pm_load_ext)

Top

```
int pm_load_ext (Promira promira, const char * app_name, PromiraLoadFlags
flags);
```

Launch an application.

Arguments

promira	handle of the connection to the Promira Serial Platform
app_name	application name to be launched
flags	enumerated values specifying the load flag See Table 3

Table 3 : `flags` enumerated types

`PM_LOAD_NO_FLAGS`	No flags.
`PM_LOAD_UNLOAD`	Unload any auto started application first.

Return Value

A Promira status code is returned with `PM_OK` on success.

Specific Error Codes

PM_APP_NOT_FOUND	There is no application with the specified name.
PM_UNABLE_TO_LOAD_APP	Unable to load the application.
PM_APP_ALREADY_LOADED	There is the auto started application. Use `PM_LOAD_UNLOAD` to forcibly load a new application.

Details

The Promira Serial Platform can have more than one application. Prior to the use of any subsystems in the application, it needs to be launched.

The Promira Serial Platform with firmware version 1.35 has three applications. `"com.totalphase.promact_is"`, `"com.totalphase.promana_espi"`, and `"com.totalphase.promact_cab"`.

Query Network Configurations (pm_query_net)

Top

```
int pm_query_net (Promira promira, PromiraNetCommand cmd, int buf_size
u08 * buf);
```

Get the Ethernet network interface properties for this Promira handle.

Arguments

promira	handle of the connection to the Promira Serial Platform
cmd	enumerated values specifying the network configuration to get. See Table 4.
buf_size	size of the array for network configuration
buf	array into which the network configuration is returned

Table 4 : `cmd` enumerated types

`PM_NET_ETH_ENABLE`	Query whether Ethernet is enabled or not.
`PM_NET_ETH_IP`	Query the IP address of Ethernet.
`PM_NET_ETH_NETMASK`	Query the netmask of Ethernet.
`PM_NET_ETH_MAC`	Query the MAC address of Ethernet.
`PM_NET_ETH_DHCP_ENABLE`	Query whether DHCP for Ethernet is enabled or not.
`PM_NET_USB_IP`	Query the IP address of Ethernet over USB.
`PM_NET_USB_NETMASK`	Query the netmask of Ethernet over USB.
`PM_NET_USB_MAC`	Query the MAC address of Ethernet over USB.

Return Value

Length of the network configuration string is returned on success.

Specific Error Codes

None.

Details

For `PM_NET_ETH_ENABLE` and `PM_NET_ETH_DHCP_ENABLE`, '1' will be returned when enabled, otherwise '0' will be returned.

Configure Network Configuration (pm_config_net)

Top

```
int pm_config_net (Promira promira, PromiraNetCommand cmd, const char *
data);
```

Configure the Ethernet network interface.

Arguments

promira	handle of the connection to the Promira Serial Platform
cmd	enumerated values specifying the network configuration to get. See Table 5
data	network configuration data

Table 5 : `cmd` enumerated types

`PM_NET_ETH_ENABLE`	Enable/Disable Ethernet.
`PM_NET_ETH_IP`	Configure the IP address of Ethernet
`PM_NET_ETH_NETMASK`	Configure the netmask of Ethernet
`PM_NET_ETH_DHCP_ENABLE`	Enable/Disable DHCP for Ethernet.
`PM_NET_ETH_DHCP_RENEW`	Renew the DHCP for Ethernet.

Return Value

A Promira status code is returned with `PM_OK` on success.

Specific Error Codes

PM_NETCONFIG_ERROR	Unable to configure network interface.
PM_INVALID_IPADDR	Invalid IP address.
PM_INVALID_NETMASK	Invalid network mask.
PM_INVALID_SUBNET	The `192.168.12.x` subnet is reserved. It is an error to configure the Ethernet interface to any address in this subnet.
PM_NETCONFIG_LOST_CONNECTION	Applying the last settings has changed the IP address of the connection. Any further access doesn't work since the connection to the device has lost and `pm_close` needs to be called to clean up the resources.

Details

In order to enable Ethernet or DHCP for Ethernet, call this function with '1' as data. In order to disable, call this with '0' as data.

Network interface can be configured by the `promira` utility (`promira.exe` or `promira`). See the section Ethernet for more detail.

Query Preferences (pm_query_pref)

Top

```
int pm_query_pref (Promira promira, const char * key, int buf_size u08 *
buf);
```

Get preferences.

Arguments

promira	handle of the connection to the Promira Serial Platform
key	key of preference
buf_size	size of the array for the value of preference
buf	array into which the value of preference is returned

Return Value

Length of the value of preference is returned on success.

Specific Error Codes

None.

Details

System firmware uses the value of **"app.autostart"** to determine which application will be loaded on boot up.

Configure Preferences (pm_config_pref)

Top

```
int pm_config_pref (Promira promira, const char * key, const char *
data);
```

Configure the preferences.

Arguments

promira	handle of the connection to the Promira Serial Platform
key	key of preference
data	preference data

Return Value

A Promira status code is returned with `PM_OK` on success.

Specific Error Codes

None.

Details

Preferences can be set by the `promira` utility (`promira.exe` or `promira`).
> promira (ip) key value

Read License (pm_read_license)

Top

```
int pm_read_license (Promira promira, int buf_size, u08 * buf);
```

Read the entire license.

Arguments

promira	handle of the connection to the Promira Serial Platform
buf_size	number of bytes in `buf`
buf	buffer to place the license string

Return Value

The length of the license in bytes.

Specific Error Codes

PM_INVALID_LICENSE	The installed license is corrupt or invalid.

Details

Pass a value of `0` for `buf` and `buf_size` to request the total length of the license.

Features (pm_features)

Top

```
int pm_features (Promira promira, const char * app, u16 features_size,
u08 * features);
```

Return the features licensed for the application.

Arguments

promira	handle of the connection to the Promira Serial Platform
app	name of the application
features_size	number of bytes in `features`
features	buffer to place the features string

Return Value

The length of the features string in bytes.

Specific Error Codes

PM_INVALID_LICENSE	The installed license is corrupt or invalid.
PM_INVALID_APP	The application name is invalid.

Details

The `features` argument will be filled with a colon separated string containing the names of features that are licensed for `app`.

Pass a value of `0` for `features` and `features_size` to request the total length of the features string. Note this length does not include the null terminating character.

Feature Value (pm_feature_value)

Top

```
int pm_feature_value (Promira promira, const char * app, const char *
feature, u16 value_size, u08 * value);
```

Return the value of a feature for the application.

Arguments

promira	handle of the connection to the Promira Serial Platform
app	name of the application
feature	name of the feature
value_size	number of bytes in `value`
value	buffer to place the value string

Return Value

The length of the value string in bytes.

Specific Error Codes

PM_INVALID_LICENSE	The installed license is corrupt or invalid.
PM_INVALID_APP	The application name is invalid.
PM_INVALID_FEATURE	The feature name is invalid.

Details

The `value` argument will be filled with a string containing the value of the `feature`. The `feature` must be one of those provided by `pm_features` for this `app`.

Pass a value of `0` for `value` and `value_size` to request the total length of the value string. Note this length does not include the null terminating character.

Feature Description (pm_feature_description)

Top

```
int pm_feature_description (Promira promira, const char * app, const char
* feature, u16 desc_size, u08 * desc);
```

Return the description of a feature for the application.

Arguments

promira	handle of the connection to the Promira Serial Platform
app	name of the application
feature	name of the feature
desc_size	number of bytes in `desc`
desc	buffer to place the description string

Return Value

The length of the description string in bytes.

Specific Error Codes

PM_INVALID_LICENSE	The installed license is corrupt or invalid.
PM_INVALID_APP	The application name is invalid.
PM_INVALID_FEATURE	The feature name is invalid.

Details

The `desc` argument will be filled with a string containing the description of the `feature`. The `feature` must be one of those provided by `pm_features` for this `app`.

Pass a value of `0` for `desc` and `desc_size` to request the total length of the description string. Note this length does not include the null terminating character.

Top

7.5 Error Codes

Table 6 : Management Error Codes

Literal Name	Value	pm_status_string() return value
PM_OK	0	ok
PM_UNABLE_TO_LOAD_LIBRARY	-1	unable to load library
PM_UNABLE_TO_LOAD_DRIVER	-2	unable to load USB driver
PM_UNABLE_TO_LOAD_FUNCTION	-3	unable to load binding function
PM_INCOMPATIBLE_LIBRARY	-4	incompatible library version
PM_INCOMPATIBLE_DEVICE	-5	incompatible device version
PM_COMMUNICATION_ERROR	-6	communication error
PM_UNABLE_TO_OPEN	-7	unable to open device
PM_UNABLE_TO_CLOSE	-8	unable to close device
PM_INVALID_HANDLE	-9	invalid device handle
PM_CONFIG_ERROR	-10	configuration error
PM_SHORT_BUFFER	-11	output buffer not large enough
PM_FUNCTION_NOT_AVAILABLE	-12	function not available
PM_APP_NOT_FOUND	-101	app not found
PM_INVALID_LICENSE	-102	invalid license
PM_UNABLE_TO_LOAD_APP	-103	unable to load app
PM_INVALID_DEVICE	-104	invalid device for license
PM_INVALID_DATE	-105	invalid date
PM_NOT_LICENSED	-106	not licensed
PM_INVALID_APP	-107	invalid app
PM_INVALID_FEATURE	-108	invalid feature
PM_UNLICENSED_APP	-109	unlicensed app
PM_APP_ALREADY_LOADED	-110	app already loaded
PM_NETCONFIG_ERROR	-201	network configuration error
PM_INVALID_IPADDR	-202	invalid IP address
PM_INVALID_NETMASK	-203	invalid netmask
PM_INVALID_SUBNET	-204	invalid subnet
PM_NETCONFIG_UNSUPPORTED	-205	unable to set network configuration
PM_NETCONFIG_LOST_CONNECTION	-206	changed, but connection has lost

8 Legal / Contact

Top

8.1 Disclaimer

All of the software and documentation provided in this manual, is copyright Total Phase, Inc. ("Total Phase"). License is granted to the user to freely use and distribute the software and documentation in complete and unaltered form, provided that the purpose is to use or evaluate Total Phase products. Distribution rights do not include public posting or mirroring on Internet websites. Only a link to the Total Phase download area can be provided on such public websites.

Total Phase shall in no event be liable to any party for direct, indirect, special, general, incidental, or consequential damages arising from the use of its site, the software or documentation downloaded from its site, or any derivative works thereof, even if Total Phase or distributors have been advised of the possibility of such damage. The software, its documentation, and any derivative works is provided on an "as-is" basis, and thus comes with absolutely no warranty, either express or implied. This disclaimer includes, but is not limited to, implied warranties of merchantability, fitness for any particular purpose, and non-infringement. Total Phase and distributors have no obligation to provide maintenance, support, or updates.

Information in this document is subject to change without notice and should not be construed as a commitment by Total Phase. While the information contained herein is believed to be accurate, Total Phase assumes no responsibility for any errors and/or omissions that may appear in this document.

Top

8.2 Life Support Equipment Policy

Total Phase products are not authorized for use in life support devices or systems. Life support devices or systems include, but are not limited to, surgical implants, medical systems, and other safety-critical systems in which failure of a Total Phase product could cause personal injury or loss of life. Should a Total Phase product be used in such an unauthorized manner, Buyer agrees to indemnify and hold harmless Total Phase, its officers, employees, affiliates, and distributors from any and all claims arising from such use, even if such claim alleges that Total Phase was negligent in the design or manufacture of its product.

Top

8.3 Contact Information

Total Phase can be found on the Internet at https://www.totalphase.com/. If you have support-related questions, please go to the Total Phase support page at https://www.totalphase.com/support/. For sales inquiries, please contact sales@totalphase.com.

©2003-2017 Total Phase, Inc.
All rights reserved.

promira-v1.35.001.pdf (500 KB)

Related articles

Advanced Cable Tester Quick Start Guide
Programming an SPI EEPROM Using Promira Serial Platform and Control Center Serial Command Line on Linux
Monitoring eSPI Messages Using Promira Platform and Data Center
Promira Serial Platform I2C/SPI Active User Manual
Programming I2C EEPROM Using Aardvark Adapter, Level Shifter Board and Control Center