Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Si3484

64-Port PoE Power Management Controller

Features

- Enables use of smaller power supplies for up to 64-port PoE systems with Si3459 and Si3454 PSE Controller ICs
- Can operate with or without a host
- Configuration save capability
- Pin-selectable SPI or UART interface
- Pin-selectable UART data rate
- Fully-compliant with IEEE 802.3-AT Types I and II
- Supports classification-based and LLDP power negotiation
- Supports individual port priority and port configuration
- Supports Power supply status from up to 3 power supplies
- 24-pin Quad flat pack package
  - 4x4 mm PCB footprint; RoHS complaint
- Extended temperature operating range (–40 to +85 °C)

Applications

- Power over Ethernet Endpoint switches and Midspans
- Supports high-power PDs, such as:
  - Pan/Tilt/Zoom security cameras
  - Wireless Access Points
  - Security and RFID systems
- Industrial automation systems
- Networked audio
- IP Phone Systems and iPBXs

Description

The Si3484 is a power manager intended for use with the Si3459 Power over Ethernet (PoE) Power Sourcing Equipment (PSE) controllers for power management of up to 64 ports with three power sources.

The Si3459 is capable of delivering over 30 W per port, which means that, in a 24- or 48-port system, a very large power supply would have to be used to avoid overload. Typically, not all ports are used at full power; so, a smaller power supply can be used along with the Si3484 power management controller.

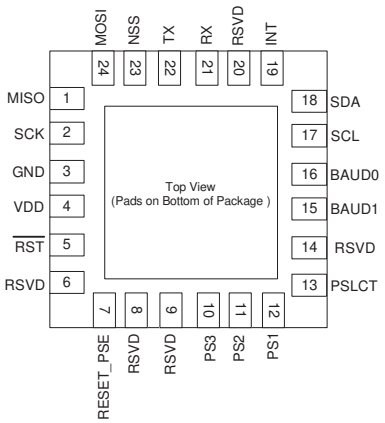
Use of the Si3484 power manager greatly simplifies system implementation of power management. The Si3484 power management controller is programmed via a SPI or UART interface to set the system power supply capacity, the port power configuration (Type 1: 15.4 W, or high-power Type 2: 30 W) ports, the port priority, the detection timing (Alternative A or Alternative B), and the fault recovery protocol. Once programmed, the configuration data can be saved, and the Si3484 can work without host intervention. Port and overall status information is available and continuously updated.

The Si3484 uses the real-time overload and current monitoring capability of the Si3459 to manage power shared among up to 64 ports. Power management is selectable between grant-based or consumption-based algorithms in order to supply power to the greatest number of ports.

In high-reliability systems, multiple power supplies are often connected to provide redundancy, which further increases the power supply monitoring requirements. The Si3484 can manage up to three power supplies automatically, enabling or disabling ports in priority order.

Pin Assignments

24-Pin QFN



See "5. Pin Descriptions" on page 45.

Functional Block Diagram

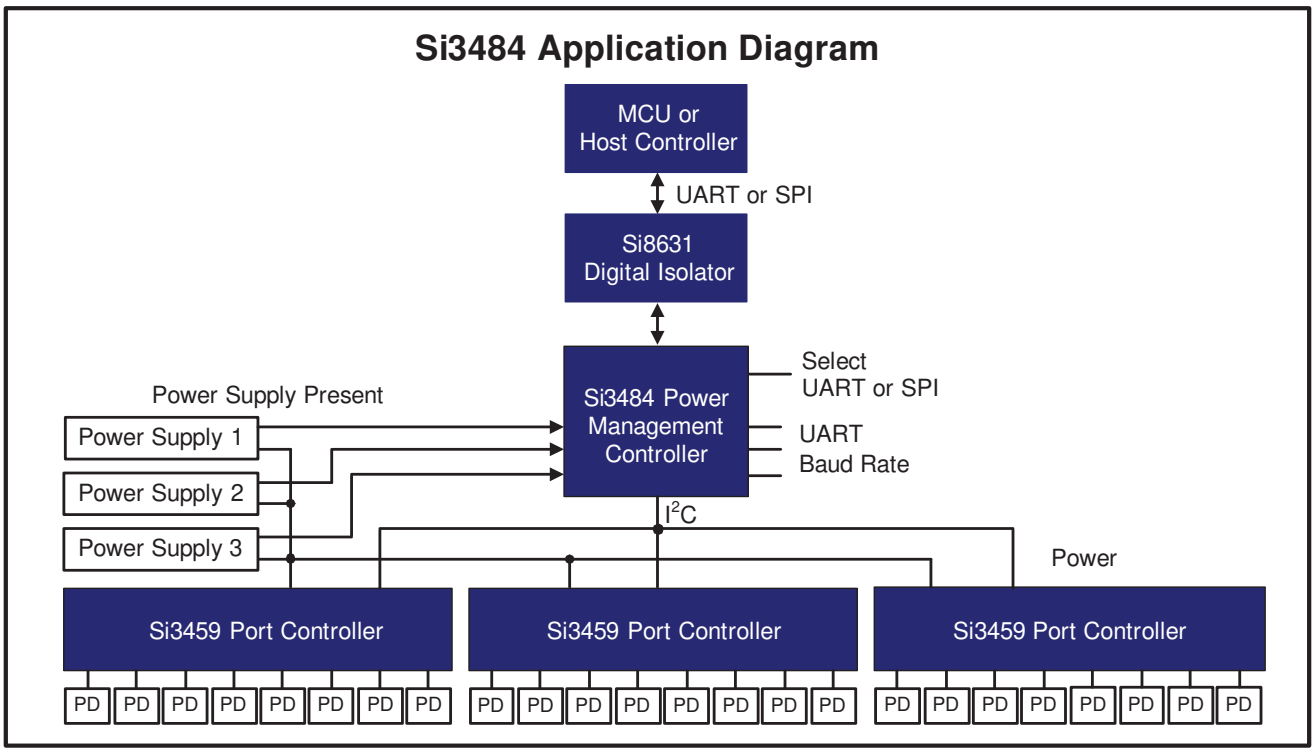


TABLE OF CONTENTS

1. Electrical Specifications

Table 1. Recommended Operating Conditions

Description	Symbol	Test Condition	Min	Typ	Max	Unit
Operating Temperature Range	T_A	No airflow	–40	—	85	°C
V_{DD} Supply Voltage	V_{DD}	All operating modes	2.7	—	3.6	V

Table 2. Absolute Maximum Ratings

Parameter	Test Condition	Min	Typ	Max	Unit
Ambient Temperature under Bias		–55	—	125	°C
Storage Temperature		–65	—	150	°C
Voltage on any I/O with Respect to GND	V_{DD}>2.2 V	–0.3	—	5.8	V
Voltage on V_{DD} with Respect to GND		–0.3	—	4.2	V

Note: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 3. Electrical Characteristics

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Input High	V_{IH}	Input pins: RST, SCK, MOSI, NSS, RX, PSn, BAUDn, SLCTIN, SCL, SDA	2.0	—	—	V
Input Low	V_{IL}		—	—	0.8	V
Input Leakage Current	I_{IL}		—	—	±1	uA
Output Low (MOSI, TX, SCL, and SDA)	V_{OL}	I_{OL} = 8.5 mA	—	—	0.6	V
Output High (MOSI, TX)	V_{OH}	I_{OH} = –3 mA	V_{DD}–0.7	—	—	V
V_{DD} Current	I_{DD}	VDD = 3.0 V* VDD = 3.6 V*	—	—	8.6 12.1	mA

***Note:** V_{DD} = 2.7 to 3.6 V, –40 to 85 °C unless otherwise noted.

Table 4. Timing Requirements

Parameter	Symbol	Min	Max	Unit
SPI Timing Requirements (See Figure 1)				
NSS Falling to First SCK Edge	T_{SE}	84	—	ns
Last SCK Edge to NSS Rising	T_{SD}	84	—	ns
NSS Falling to MISO Valid	T_{SEZ}	—	168	ns
NSS Rising to MISO High Z	T_{SDZ}	—	168	ns
SCK High Time	T_{CKH}	210	—	ns
SCK Low Time	T_{CKL}	210	—	ns
MOSI Valid to SCK Sample Edge	T_{SIS}	84	—	ns
SCK Sample Edge to MOSI Change	T_{SIH}	84	—	ns
SCK Shift Edge to MISO Change	T_{SCH}	—	168	ns
Maximum SPI Clock Speed	F_{MAX}	—	1	MHz
UART Requirements (See Figure 2)				
Deviation of Tx Transmit Speed from Pin-programmed Value	ΔFTx	–3	+3	%
Deviation of Rx receive Speed from Pin-programmed Value	ΔFRx	–4	+4	%

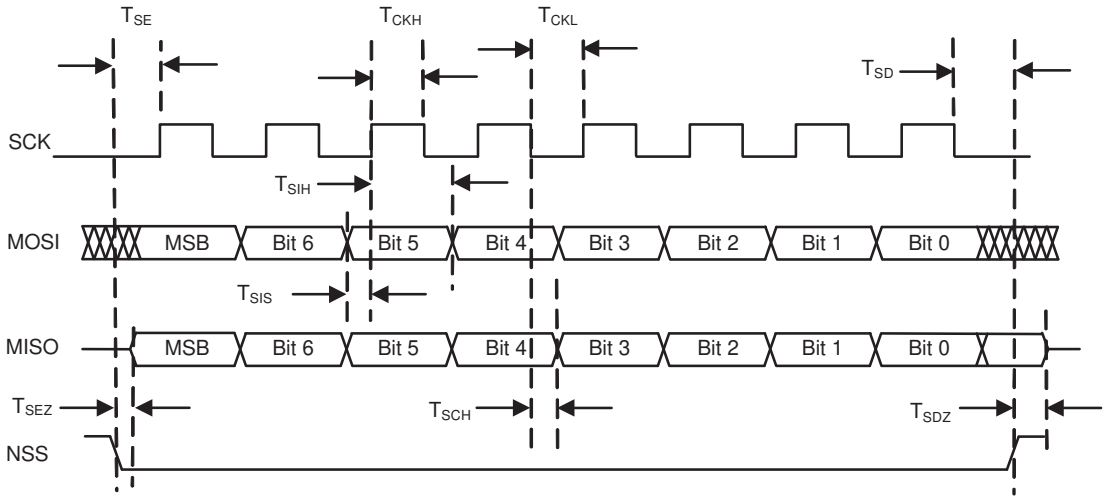


Figure 1. SPI Timing Diagram

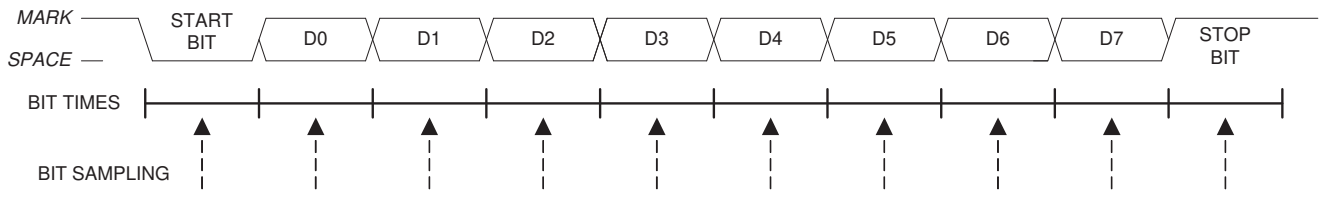


Figure 2. UART Timing Diagram

2. Functional Description

The Si3484 Power Management Controller takes the role of the central controller in a Silicon Labs Power over Ethernet (PoE) system. In a PoE system, power is provided by one or more power supplies and is consumed by one or more powered devices (PDs). The Si3484 decides which of the PDs can have power and monitors the amount of power consumed by each.

A host microcontroller unit (MCU) can configure the Si3484 and can query the status of the PDs and the power supplies. The Si3484 stores its configuration in internal flash memory. A host MCU uses a Universal Asynchronous Receiver Transmitter (UART) or a Serial Peripheral Interface (SPI) to communicate with the Si3484. Pins on the Si3484 select which host interface to use and which baud rate to use for the UART interface.

Power supplies may be inserted into bays. The Si3484 supports a system with up to three bays. Power supplies may be inserted or removed from the bays at any time. Each bay provides a signal to the Si3484 that indicates if a power supply is present and operational in the bay. The outputs of the power supplies are ganged together to provide a single power source for the system.

The Si3484 manages a collection of Si3459 Port Controllers. The Si3484 supports a system with up to 8 Si3459s. Each Si3459 has eight ports; so, a system may have up to 64 ports. The Si3459 performs low-level port functions, such as detecting and classifying PDs. The Si3484 has a global view of the system and manages power across all ports.

PDs are connected to ports on the Si3459s. PDs may be connected or disconnected from the ports at any time. When a PD is connected to a port, then the PD requests power from the port. The Si3484 determines the amount of power requested from the classification of the PD. If there is enough power remaining, the Si3484 grants the request; otherwise, the Si3484 denies the request.

The host may configure an optional power limit for each port. A power limit restricts the amount of power that the Si3484 grants to a port. If a power request is greater than the power limit, the Si3484 does not fully grant the request, but only grants the amount of the power limit.

The Si3484 supports Link Layer Discovery Protocol (LLDP) agents in the host. An LLDP agent can call a routine in the Si3484 to dynamically adjust the amount of power granted to a PD during the course of a connection.

Several PDs may be connected to a PoE system. The Si3484 may have granted different amounts of power to each PD, and each PD may be consuming different amounts of power. If a PD consumes more power than it is granted (port overload), the Si3484 turns off the PD.

There are two approaches that the Si3484 can take when granting requests for power. The granting policy can be grant-based or it can be consumption-based.

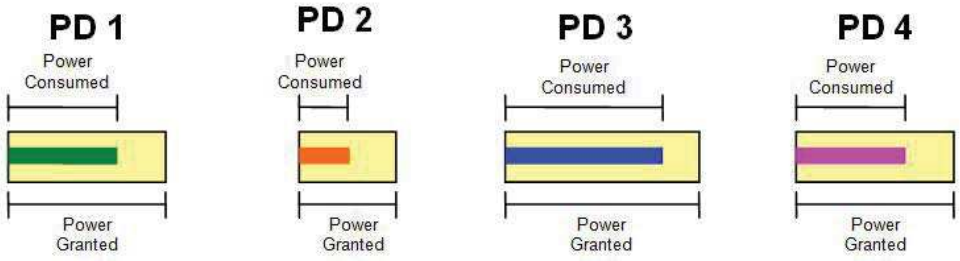


Figure 3. Powered Devices Example

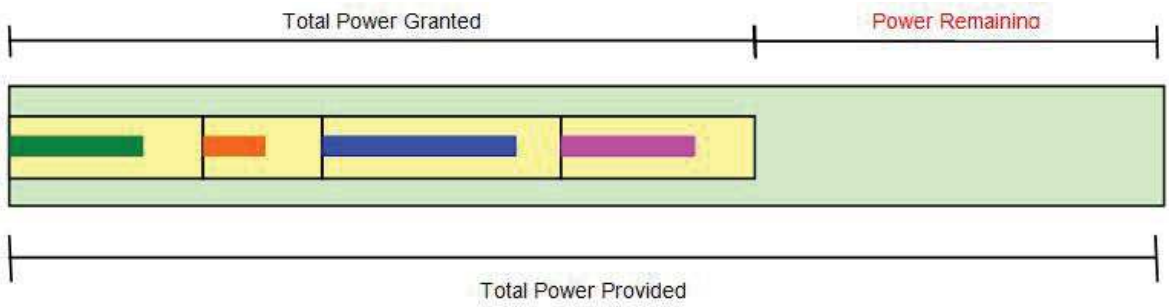


Figure 4. Grant Based Power Management

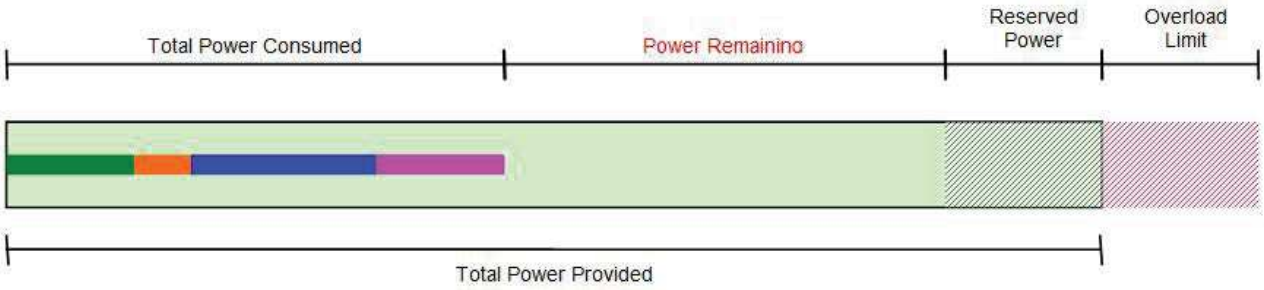


Figure 5. Consumption-Based Power Management

If the granting policy is grant based, then the power remaining for new grants is the total ungranted power. The power remaining is the total power provided minus the total power granted.

The problem with this approach is that much of the provided power is unused because PDs often do not consume all of their granted power.

If the granting policy is consumption-based, then the power remaining for new grants is the total unconsumed power. The power remaining is the total power provided minus the total power consumed (excluding the reserved power). This approach uses more of the provided power, but there is a possibility that the system may consume more power than the power provided (system overload).

To avoid system overloads caused by momentary surges in power consumption, the host can specify that a certain amount of power be held in reserve. The Si3484 does not use the reserved power when granting new requests.

Most power supplies can tolerate a limited amount of overload for a short duration. The host specifies the overload limit of the power supplies to the Si3484. If a system overload is less than the overload limit, the Si3484 turns off ports, one at a time in priority order, until the system is no longer overloaded. If a system overload is greater than the overload limit (severe overload), the Si3484 immediately turns off all low-priority ports. If the system is still overloaded, the Si3484 turns off additional ports, one at a time in priority order, until the system is no longer overloaded. A severe overload is usually caused by removing a power supply.

2.1. Host Interface

The Si3484 has a UART interface and an SPI interface for communicating with the host MCU, but only one interface is used at a time. The PSLCT (protocol select) pin selects which interface is used.

2.1.1. UART Interface

If the PSLCT pin is tied high, then the Si3484 uses the UART interface to communicate with the host MCU. The Si3484 uses the TX and RX pins to send and receive serial data. The BAUD0 and BAUD1 pins select the baud rate for the UART interface.

Table 5. Baud Rates

BAUD1	BAUD0	Baud Rate (bps)
L	L	19200
L	H	38400
H	L	57600
H	H	115200

The UART interface uses eight data bits, no parity, and one stop bit.

2.1.2. SPI Interface

If the PSLCT pin is tied low, then the Si3484 uses the SPI interface to communicate with the host MCU. The Si3484 is an SPI slave device. Therefore, it receives data on the MOSI pin and sends data on the MISO pin. The host MCU drives the NSS and SCK pins.

The SPI interface uses an active-high clock (CKPOL = 0). The clock line is low in the idle state, and the leading edge of the clock goes from low to high. The SPI interface samples the data on the leading edge of the clock (CKPHA = 0). The SPI interface transfers the most-significant bit first, and the maximum bit rate is 1 Mbps.

2.2. Hardware Only Mode

The host interface (SPI or UART) and the UART baud rate are pin-configured. The Si3484 reads the pin configuration at power up, and it cannot be changed after power up. The hardware designer only needs to decide which interface to use and, if UART is selected, which BAUD rate to use.

In general, the host interface must be electrically isolated from the host MCU using an appropriate electrical isolator for either SPI or UART signals as well as power supply status signals as needed.

The Si3484 backs up its configuration to internal flash memory. Once the Si3484 is configured, it is possible to disconnect the host interface and use the Si3484 without a host MCU.

3. Serial Packet Protocol

The Si3484 contains the Power Manager component and the interface to the Power Manager is a collection of routines known as the Power Manager application programming interface (API).
The Power Manager API is described later in this manual. The host MCU should contain a Serial Packet Client, which calls the routines in the Power Manager API to get status information and configure and control the Power Manager.

The Serial packet protocol (SPP) is a remote procedure call (RPC) mechanism that allows a Serial Packet client to call routines in the Power Manager. The Serial Packet Protocol is implemented by a Serial Packet Client in the host MCU and the Serial Packet Server in the Si3484. The Serial Packet Client should be implemented by the user in the host MCU. Silicon Labs has reference code available; please contact Silicon Labs for further information.

The Serial Packet Server receives a packet from a Serial Packet Client and then calls the specified routine in the Power Manager. When the Power Manager routine returns, the Serial Packet Server sends a packet back to the Serial Packet Client.

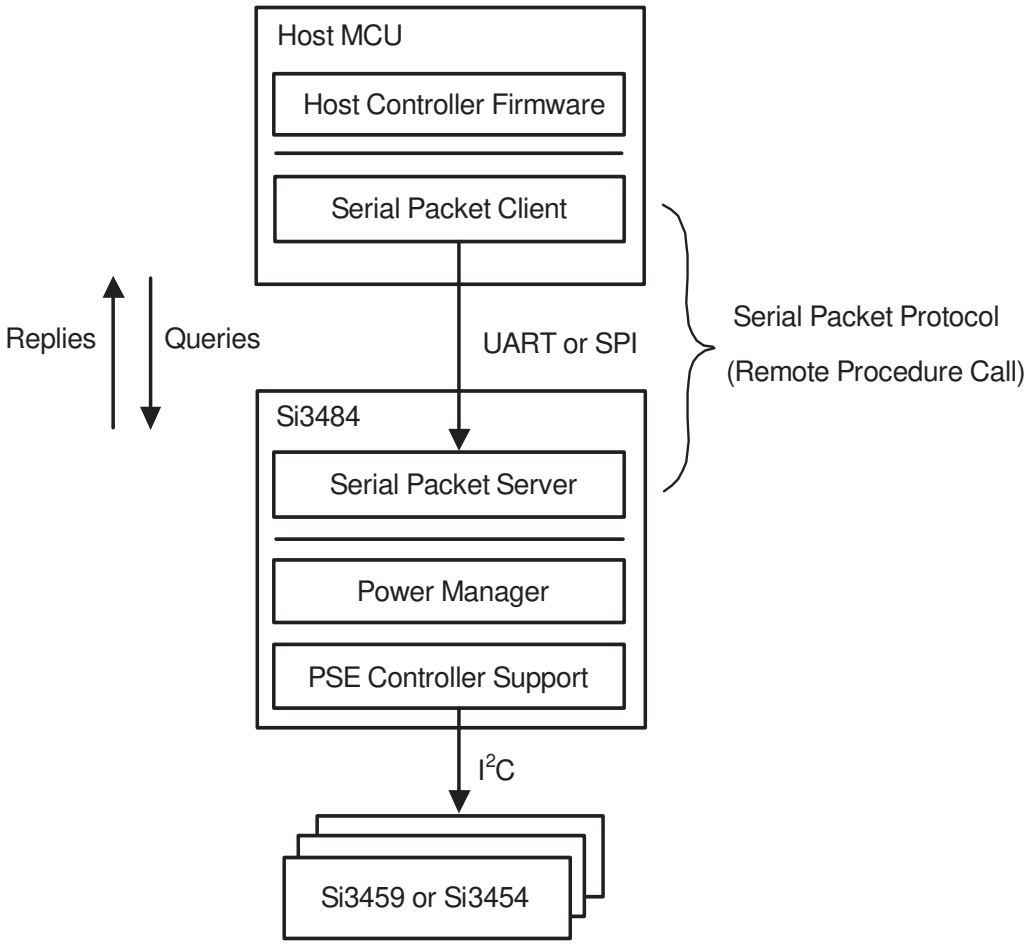


Figure 6. Serial Packet Protocol

3.1. Packet Format

A packet is a sequence of fields sent together as a unit. Figure 7 shows the SPP packet format.

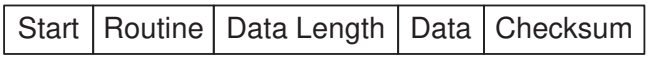


Figure 7. Packet Format

Each field is a single byte except for the Data field. The Data field length may be from zero to 255 bytes.

3.1.1. Start Field

The Start field marks the beginning of a packet and always contains the Start-of-Packet (SOP) character (0xAC). If data is lost on the host interface, the Serial Packet Server and the Serial Packet Client use the Start field to resynchronize. A “receive packet” routine starts by receiving and discarding bytes until the SOP character is found.

3.1.2. Checksum Field

The Checksum field is used to verify that the packet was not corrupted during transmission. The sender of a packet calculates the checksum and writes it into the Checksum field. The receiver of a packet verifies that the checksum is correct. The Checksum field should contain the value such that all the bytes in the packet, except for the Start field, add up to zero.

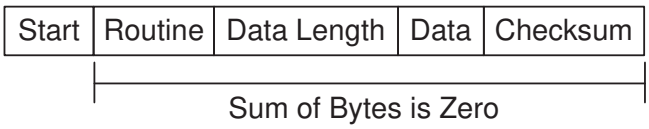


Figure 8. Packet Checksum

To calculate the checksum, the sender uses an 8-bit variable to sum up the bytes of the Routine field through the end of the Data field. The sender adds one to the one's complement of this sum and stores the result in the Checksum field.

$$\text{Checksum} = (\sim\text{Sum}) + 1$$

To verify the checksum, the receiver uses an 8-bit variable to sum up the bytes of the Routine field through the Checksum field. The sum should be zero.

3.1.3. Routine Field

The Routine field identifies a routine in the Power Manager API.

The client uses the Routine field to specify which routine to call. The client should verify that the Routine field in a received packet matches the Routine field in the sent packet. Definitions of routines can be found in “4. Power Manager API” .

3.1.4. Data Length Field

The DataLength field specifies the number of bytes in the Data field. The number of bytes may be from zero to 255.

3.1.5. Data Field

The Data field is used to pass data to and from the Si3484. The Data field may contain four different types of data:

- Parameters
- System Information
- Port Information
- Events

The Data field has a different format for each type of data. The format of commands issued by the host to the power manager is fixed, but the format of the returned values may be of four different types: Parameters, System Information, Port Information, and Events.

To elaborate, in all packets issued by the host, the Data Field has the Parameters format. Most of the returned packet are also in the Parameters format, the only exceptions being the packets that are received back after calling

the RTN_GETSYSTEMINFO, RTN_GETPORTINFO, and RTN_GETEVENTS routines. The Data Fields for these return packets are in the System Information format, Port Information format, and Events format, respectively.

3.1.5.1. Parameters Format

The Parameters format of the Data field is used to pass parameters to Power Manager routines. In most cases, the Parameters format is also used to return data from the routines.

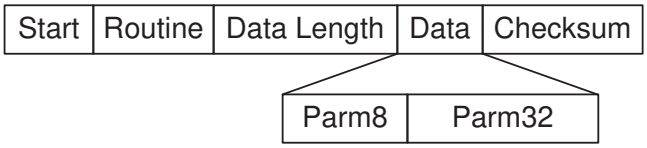


Figure 9. Parameters Format

The Parameters format has an 8-bit Parm8 field followed by a 32-bit Parm32 field (see Table 6). Depending on the routine being called, Parm8, Parm32, or both fields are used. Sometimes, neither field is used. However, both fields are always sent and received. The DataLength field contains five.

Table 6. Use of Parameters

Routine	Parameters in Query Packet		Parameters in Reply Packet	
	Parm8	**Parm32***	**Parm8**	**Parm32***
Get System Status			SystemStatus	
Get System Info			Uses System Information Format	
Get Total Power Consumed				PowerConsumed
Get Total Power Granted				PowerGranted
Get Total Power Provided				PowerProvided
Get Port Count			PortCount	
Get Port Status	Port		PortStatus	
Get Port Info	Port		Uses Port Information Format	
Get Port Priority Status	Port		PortPriorityStatus	
Get Port Power Consumed	Port			PowerConsumed
Get Port Power Granted	Port			PowerGranted
Get Port Power Requested	Port			PowerRequested
Get Port Power Available	Port			PowerAvailable
Reset System			Result	
Restore Factory Defaults				
Store Configuration				
Set Port Control	Port	Control	Result	
Adjust Port Power	Port	PortPower	Result	
Set Power Provided	PowerSupply	PowerProvided	Result	

***Note:** The Parm32 field is big endian; therefore, the most significant byte is first.

Table 6. Use of Parameters (Continued)

Routine	Parameters in Query Packet		Parameters in Reply Packet	
	Parm8	Parm32*	Parm8	Parm32*
Get Power Provided	PowerSupply			PowerProvided
Set Reserved Power	ReservedPower		Result	
Get Reserved Power			ReservedPower	
Set Overload Limit	OverloadLimit		Result	
Get Overload Limit			OverloadLimit	
Set Granting Policy	GrantingPolicy		Result	
Get Granting Policy			GrantingPolicy	
Set Retry Policy	RetryPolicy		Result	
Get Retry Policy			RetryPolicy	
Set Port Enable	Port	Enable	Result	
Get Port Enable	Port		Enable	
Set Port Capability	Port	Capability	Result	
Get Port Capability	Port		Capability	
Set Port Midspan	Port	Location	Result	
Get Port Midspan	Port		Location	
Set Port Priority	Port	Priority	Result	
Get Port Priority	Port		Priority	
Set Port Legacy Support	Port	Legacy	Result	
Get Port Legacy Support	Port		Legacy	
Set Port Power Limit	Port	PowerLimit	Result	
Get Port Power Limit	Port			PowerLimit
Get Power Supply Status	PowerSupply		Status	
Get Events			Uses Events Format	
Get System Time				System Time in ms
Set Power Inflation	Power Inflation			
Get Power Inflation			Power Inflation	
Set Soft Start Policy	Soft Start Policy			
Get Soft Start Policy			Soft Start Policy	

***Note:** The Parm32 field is big endian; therefore, the most significant byte is first.

3.1.5.2. System Information Format

The System Information format of the Data field is used to return system information to the client. System information is returned after calling the RTN_GETSYSTEMINFO routine.

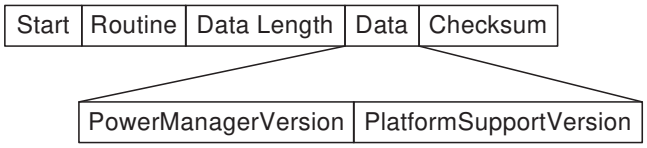


Figure 10. System Information Format

The System Information format has a PowerManagerVersion field followed by a PlatformSupportVersion field. Both of these fields are eight bytes long and contain a version string that is a zero-terminated ASCII string. A version string may be from one to seven characters long. The Routine field contains RTN_GETSYSTEMINFO, and the DataLength field contains 16.

3.1.5.3. Port Information Format

The Port Information format of the Data field is used to return port information to the client. Port information is returned after calling the RTN_GETPORTINFO routine.

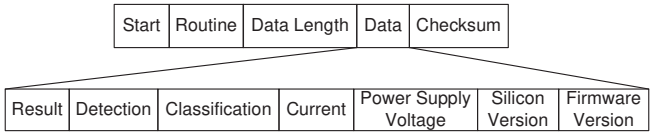


Figure 11. Port Information Format

The Port Information format is a sequence of fields as shown above. For more information, read the description of the RTN_GETPORTINFO routine in the Power Manager API Section. The Routine field contains RTN_GETPORTINFO, and the DataLength field contains 17.

The Result field contains the return code from the RTN_GETPORTINFO routine, and, if Result is not SUCCESS (0), the remaining fields should be ignored.

The Current and PowerSupplyVoltage fields are big endian. Therefore, the most significant byte comes first.

3.1.5.4. Events Format

The Events format of the Data field is used to return events to the client. Events are returned after calling the RTN_GETEVENTS routine.

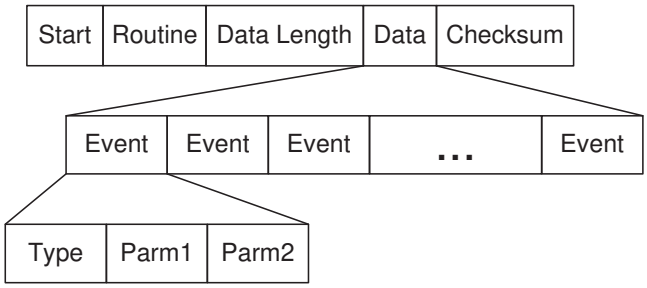


Figure 12. Events Format

In the Si3484, the Serial Packet Server internally receives events from the Power Manager and stores them in a circular event queue. If the event queue becomes full, newer events overwrite older events.

If a client wishes to receive events, it should periodically get the events from the Serial Packet Server. The client gets the events by sending a packet with the Routine field set to RTN_GETEVENTS. The Serial Packet Server returns all the events from the event queue in a single packet with the Data field in the Events format.

The Data field does not have a fixed length. The length of the Data field depends on the number of events that are returned. An event is three bytes long; so, the number of events in the Data field is DataLength divided by three. If there are no events to return, then DataLength is zero, and the Data field is empty. The maximum number of events that can be returned is 72.

3.1.6. Serial Packet Formats

All four packet formats must use the same “Parameters” packet format when the host issue a transfer even if the return format is System Info, Port Info or Event.

3.2. SPP Error Handling

There are many reasons why a client may not receive back a packet. Perhaps the Si3484 is not running or perhaps the serial data was corrupted or lost during transmission (in either direction). In any case, it is not prudent for a Serial Packet Client to call a serial receive routine that blocks forever until data is received. If the serial receive routine does not have a timeout option, the client should not call the receive routine unless it knows that received data is available. If a client does not receive a packet within one second of sending a packet, then the client should assume that there has been a communications error. The client should resend the original packet or simply give up (but do not wait forever to receive a packet).

When the Serial Packet Server receives a packet, it validates the packet. If the checksum is bad or the Routine field is invalid, the Serial Packet Server ignores the packet and does not send back a packet in response. After one second, the client should realize that a packet has not been received and should resend the original packet.

The Si3484 checks the configuration every 30 seconds to see if it has changed. If the configuration has changed, the Si3484 backs up the configuration to internal flash memory. While the Si3484 is writing to flash memory, it cannot send or receive packets on the host interface. If a host MCU sends a packet to the Si3484 while it is backing up the configuration, the packet is lost. If a host MCU does not receive a packet back within one second, the host MCU should resend the original packet.

4. Power Manager API

User Interface components call the routines in the Power Manager API to get status information and configure and control the Power Manager. The Power Manager API has routines for:

- Management
- System Status
- Port Status
- System Control
- Port Control
- System Configuration
- Port Configuration
- Power Supply Status
- Events

Output packet format is ‘Parameters Format’ in all cases. Input packet format depends on the routine.

Maximum of the ‘Port’ parameter of the routines (where applicable) is 64 ports, but only ports available in the system give a valid result.

Some functions can emit error codes. Descriptions of the error codes can be found in "4.9. Return Codes" on page 44

Values in the “**Symbol**” column of the tables are recommended names for constant values.

To build a valid query packet, the user must specify the routine and provide the necessary parameters. If the parameter is indicated as “None”, the serial packet server does not rely on the passed value, so the recommended value is 0. In case of receiving reply packets, the parameters indicated as “None” should be ignored by the host. The serial packet server will echo back the routine name in the reply packet, which can be used along with the checksum value to check for consistency.

Table 7 contains routines available through the serial packet protocol.

Table 7. Power Manager Routines

Routine	Value	Symbol
Get System Status	1	RTN_GETSYSTEMSTATUS
Get System Info	2	RTN_GETSYSTEMINFO
Get Total Power Consumed	3	RTN_GETTOTALPOWERCONSUMED
Get Total Power Granted	4	RTN_GETTOTALPOWERGRANTED
Get Total Power Provided	5	RTN_GETTOTALPOWERPROVIDED
Get Port Count	6	RTN_GETPORTCOUNT
Get Port Status	7	RTN_GETPORTSTATUS
Get Port Info	8	RTN_GETPORTINFO
Get Port Priority Status	9	RTN_GETPORTPRIORITYSTATUS
Get Port Power Consumed	10	RTN_GETPORTPOWERCONSUMED
Get Port Power Granted	11	RTN_GETPORTPOWERGRANTED
Get Port Power Requested	12	RTN_GETPORTPOWERREQUESTED

Table 7. Power Manager Routines (Continued)

Routine	Value	Symbol
Get Port Power Available	13	RTN_GETPORTPOWERAVAILABLE
Reset System	14	RTN_RESETSYSTEM
Restore Factory Defaults	15	RTN_RESTOREFACTORYDEFAULTS
Set Port Control	16	RTN_SETPORTCONTROL
Adjust Port Power	17	RTN_ADJUSTPORTPOWER
Set Power Provided	18	RTN_SETPOWERPROVIDED
Get Power Provided	19	RTN_GETPOWERPROVIDED
Set Reserved Power	20	RTN_SETRESERVEDPOWER
Get Reserved Power	21	RTN_GETRESERVEDPOWER
Set Overload Limit	22	RTN_SETOVERLOADLIMIT
Get Overload Limit	23	RTN_GETOVERLOADLIMIT
Set Granting Policy	24	RTN_SETGRANTINGPOLICY
Get Granting Policy	25	RTN_GETGRANTINGPOLICY
Set Retry Policy	26	RTN_SETRETRYPOLICY
Get Retry Policy	27	RTN_GETRETRYPOLICY
Set Port Enable	28	RTN_SETPORTENABLE
Get Port Enable	29	RTN_GETPORTENABLE
Set Port Capability	30	RTN_SETPORTCAPABILITY
Get Port Capability	31	RTN_GETPORTCAPABILITY
Set Port Midspan	32	RTN_SETPORTMIDSPAN
Get Port Midspan	33	RTN_GETPORTMIDSPAN
Set Port Priority	34	RTN_SETPORTPRIORITY
Get Port Priority	35	RTN_GETPORTPRIORITY
Set Port Legacy Support	36	RTN_SETPORTLEGACYSUPPORT
Get Port Legacy Support	37	RTN_GETPORTLEGACYSUPPORT
Set Port Power Limit	38	RTN_SETPORTPOWERLIMIT
Get Port Power Limit	39	RTN_GETPORTPOWERLIMIT
Set Power Supply Status	40	RTN_SETPOWERSUPPLYSTATUS
Get Power Supply Status	41	RTN_GETPOWERSUPPLYSTATUS

Table 7. Power Manager Routines (Continued)

Routine	Value	Symbol
Get Events	42	RTN_GETEVENTS
Store Configuration	43	RTN_STORECONFIG
Set Soft Start Policy	45	RTN_SETSOFTSTARTPOLICY
Get Soft Start Policy	46	RTN_GETSOFTSTARTPOLICY
Get System Time	47	RTN_GETSYSTEMTIME
Set Power Inflation	48	RTN_SETPOWERINFLATION
Get Power Inflation	49	RTN_GETPOWERINFLATION

4.1. System Status

The System Status routines allow a User Interface component to get the following information:

- System Status
- System Info
- Total Power Consumed
- Total Power Granted
- Total Power Provided

4.1.1. Get System Status

Get the status of the system.

Routine:

`RTN_GETSYSTEMSTATUS`

Query data:

Parm8: `None`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `System status`

Parm32: `None`

The system status is the overall status of the system. A negative system status value is an error that is not specific to a particular port.

Table 8. System Status Values

Status	Value	Symbol
OK	0	STATUS_SYSTEM_OK
Initialization Failed	–1	STATUS_SYSTEM_INIT_FAIL
Under Voltage	–2	STATUS_SYSTEM_UNDER_VOLT
Over Temperature	–3	STATUS_SYSTEM_OVER_TEMP
Communications Lost	–4	STATUS_SYSTEM_COMM_LOST

4.1.2. Get System Info

Get information about the system.

Routine:

RTN_GETSYSTEMINFO

Query data:

Parm8: None

Parm32: None

Reply packet format:

System Information

Reply data:

Bytes 0..7 Power Manager Version (zero-terminated string)

Bytes 8..15 Platform Support Version (zero-terminated string)

Return strings contain the version of the Power Manager and the version of the Platform Support component as zero-terminated strings.

4.1.3. Get Total Power Consumed

Get the power consumed by all PDs.

Routine:

RTN_GETTOTALPOWERCONSUMED

Query data:

Parm8: None

Parm32: None

Reply packet format:

Parameters

Reply data:

Parm8: None

Parm32: Total power consumed in mW

4.1.4. Get Total Power Granted

Get the power granted to all PDs.

Routine:

RTN_GETTOTALPOWERGRANTED

Query data:

Parm8: None

Parm32: None

Reply packet format:

Parameters

Reply data:

Parm8: None

Parm32: Total power granted in mW

4.1.5. Get Total Power Provided

Get the power provided by all power supplies

Routine:

`RTN_GETTOTALPOWERPROVIDED`

Query data:

Parm8: `None`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `None`

Parm32: `Total power provided in mW`

4.2. Port Status

The Port Status routines allow a User Interface component to get the following information:

- Port Count
- Port Status
- Port Info
- Port Priority Status
- Port Power Consumed
- Port Power Granted
- Port Power Requested
- Port Power Available

4.2.1. Get Port Count

Get the number of ports in the system.

Routine:

`RTN_GETPORTCOUNT`

Query data:

Parm8: `None`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `Number of ports in the system`

Parm32: `None`

When the Power Manager starts up, it discovers the number of ports in the system by searching for port controllers.

4.2.2. Get Port Status

Get the status of a port.

Routine:

`RTN_GETPORTSTATUS`

Query data:

Parm8: `Port number`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `Port status value or an error code`

Parm32: `None`

Table 9. Port Status Values

Status	Value	Symbol	Description
Disabled	0	STATUS_PORT_DISABLED	The port is off because it is not allowed to turn on.
Powered On	1	STATUS_PORT_POWERED_ON	A PD is connected and receiving power.
Powered Off	2	STATUS_PORT_POWERED_OFF	The port is off because a PD is not connected.
Denied	3	STATUS_PORT_DENIED	The port is off because there is not enough power remaining to grant the power request.
Blocked	4	STATUS_PORT_BLOCKED	The port is off because of a port overload.
Forced On	5	STATUS_PORT_FORCED_ON	The user forced the port on.
Forced Off	6	STATUS_PORT_FORCED_OFF	The user forced the port off.

If a port is blocked, then the PD consumed more power than it was granted (port overload), and the retry policy is “retry after reconnect”. To remove the block, the user must physically disconnect the PD from the port. Another way to remove the block is to disable and then re-enable the port.

4.2.3. Get Port Info

Get low-level port information.

Routine:

`RTN_GETPORTINFO`

Query data:

Parm8: `Port number`

Parm32: `None`

Reply packet format:

`Port Information`

Reply data:

Byte 0: `Result`

Byte 1: `Detection value` (see table 11.)

Byte 2: `Classification value` (see table 12.)

Byte 3..4: `Port current in mA`, 16bit, MSB first

Byte 5..6: `Power supply voltage in mV`, 16bit, MSB first

Byte 7..8: `PSE silicon version`

Byte 9..14: `PSE firmware version`

Table 10. Detect Values

Detection Result	Value	Symbol
Unknown	0	DETECT_UNKNOWN
Short	1	DETECT_SHORT
Low	3	DETECT_LOW
Good	4	DETECT_GOOD
High	5	DETECT_HIGH
Open	6	DETECT_OPEN

Table 11. Classification Values

Classification Result	Value	Symbol
Unknown	0	CLASS_UNKNOWN
Class 1	1	CLASS_1
Class 2	2	CLASS_2
Class 3	3	CLASS_3
Class 4	4	CLASS_4
Unequal fingers	5	CLASS_UNEQ_FINGERS
Class 0	6	CLASS_0
Overload	7	CLASS_OVERLOAD

4.2.4. Get Port Priority Status

Get the priority status of a port.

Routine:

`RTN_GETPORTPRIORITYSTATUS`

Query data:

Parm8: `Port number`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `Port priority status`

Parm32: `None`

Table 12. Port Priority Status Values

Status	Value	Symbol
Low	0	PRIORITY_LOW
High	1	PROIRITY_HIGH
Forced	2	PRIORITY_FORCED
Critical	3	PRIORITY_CRITICAL

The priority status of a port is the currently-active priority and may be different than the configured priority of the port. If a port is forced on or off and the configured priority is low or high, the priority status is elevated to the forced priority. If a forced port is returned to automatic control, the Power Manager returns the priority status to the configured priority.

4.2.5. Get Port Power Consumed

Get the power that a PD is currently using.

Routine:

`RTN_GETPORTPOWERCONSUMED`

Query data:

Parm8: `Port`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `None`

Parm32: `Port power consumed in mW or an error code`

4.2.6. Get Port Power Granted

Get the power that is allocated to a PD.

Routine:

`RTN_GETPORTPOWERGRANTED`

Query data:

Parm8: `Port`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `None`

Parm32: `Port power granted in mW or an error code`

4.2.7. Get Port Power Requested

Get the power that is requested by a PD.

Routine:

`RTN_GETPORTPOWERREQUESTED`

Query data:

Parm8: `Port`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `None`

Parm32: `Port power requested in mW or an error code`

4.2.8. Get Port Power Available

Get the power that is available for a PD.

Routine:

`RTN_GETPORTPOWERAVAILABLE`

Query data:

Parm8: `Port`

Parm32: `None`

Reply packet format:

`Parameters`

Reply data:

Parm8: `None`

Parm32: `Port power available in mW or an error code`

An LLDP agent calls this routine to determine maximum power that the power manager can provide for a port. If a port has a power limit, then the power limit is returned. If a port does not have a power limit and the port can supply high power, then 40 W (maximum power) is returned; otherwise, 15.4 W (low power) is returned.