# Power Management Controller

## Features

- Enables use of smaller power supplies for up to 64-port PoE systems with Si3459 and Si3454 PSE interface 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 Si3483 is a power manager intended for use with the Si3459 Power over Ethernet (PoE) 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 Si3483 power management controller.

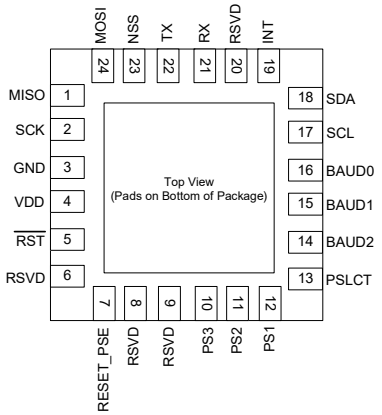
Use of the Si3483 power manager greatly simplifies system implementation of power management. The Si3483 power management controller is programmed via a SPI or UART interface to set the system power supply capacity, the port power configuration (Type 1: 15.4 W, or high-power Type 2: 30 W) ports, the 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 Si3483 can work without host intervention. Port and overall status information is available and continuously updated.

The Si3483 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 Si3483 can manage up to three power supplies automatically, enabling or disabling ports in priority order.

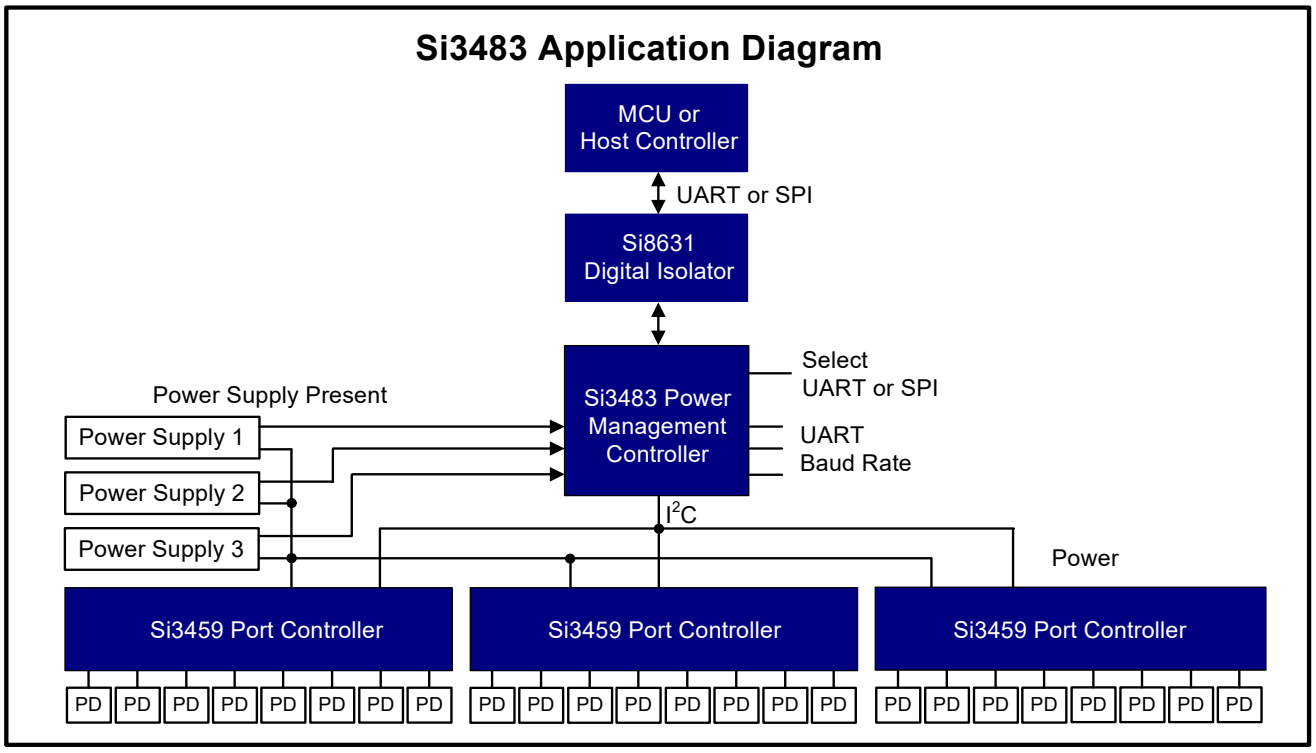
**Pin Assignments**

**24-Pin QFN**




See "5. Pin Descriptions" on page 43.

## Functional Block Diagram



# TABLE OF CONTENTS


| Section | Page |
|---|---|
| **1. Electrical Specifications** | **4** |
| **2. Functional Description** | **6** |
| 2.1. Host Interface | 7 |
| 2.2. Hardware Only Mode | 8 |
| **3. Serial Packet Protocol** | **9** |
| 3.1. Packet Format | 10 |
| 3.2. SPP Error Handling | 14 |
| **4. Power Manager API** | **15** |
| 4.1. System Status | 18 |
| 4.2. Port Status | 21 |
| 4.3. System Control | 26 |
| 4.4. Port Control | 27 |
| 4.5. System Configuration | 29 |
| 4.6. Port Configuration | 34 |
| 4.7. Power Supply Status | 40 |
| 4.8. Events | 41 |
| 4.9. Return Codes | 42 |
| **5. Pin Descriptions** | **43** |
| **6. Ordering Guide** | **45** |
| **7. Package Outline: 24-Pin QFN** | **46** |
| **8. PCB Land Pattern** | **48** |
| **9. Top Marking Diagram** | **50** |
| **Document Change List** | **51** |
| **Contact Information** | **52** |

# 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*<br>VDD = 3.6 V* | — | — | 8.6<br>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 | % |

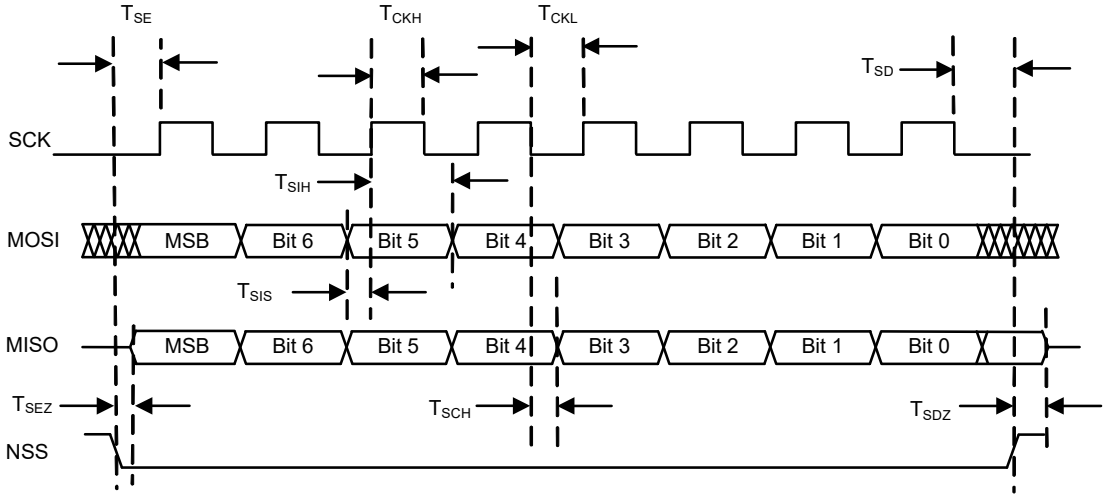



**Figure 1. SPI Timing Diagram**

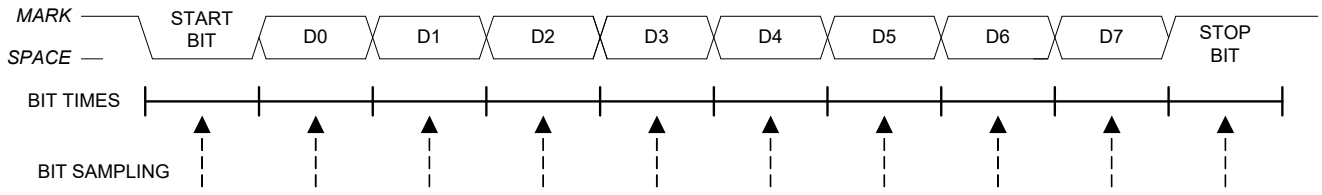



**Figure 2. UART Timing Diagram**

## 2. Functional Description

The Si3483 Power Management Controller takes the role of the central controller in a Skyworks Solutions 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 Si3483 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 Si3483 and can query the status of the PDs and the power supplies. The Si3483 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 Si3483. Pins on the Si3483 select which host interface to use and which baud rate to use for the UART interface.

Power supplies may be inserted into bays. The Si3483 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 Si3483 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 Si3483 manages a collection of Si3459 Port Controllers. The Si3483 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 Si3483 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 Si3483 determines the amount of power requested from the classification of the PD. If there is enough power remaining, the Si3483 grants the request; otherwise, the Si3483 denies the request.

The host may configure an optional power limit for each port. A power limit restricts the amount of power that the Si3483 grants to a port. If a power request is greater than the power limit, the Si3483 does not fully grant the request, but only grants the amount of the power limit.

The Si3483 supports Link Layer Discovery Protocol (LLDP) agents in the host. An LLDP agent can call a routine in the Si3483 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 Si3483 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 Si3483 turns off the PD.

There are two approaches that the Si3483 can take when granting requests for power. The granting policy can be grant-based or it can be consumption-based.

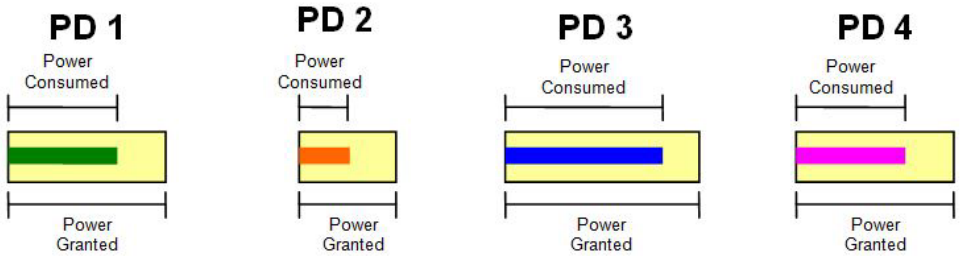



**Figure 3. Powered Devices Example**

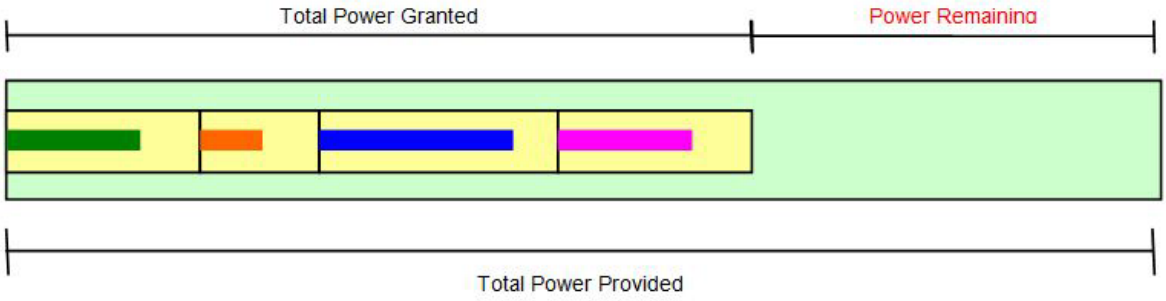



**Figure 4. Grant Based Power Management**

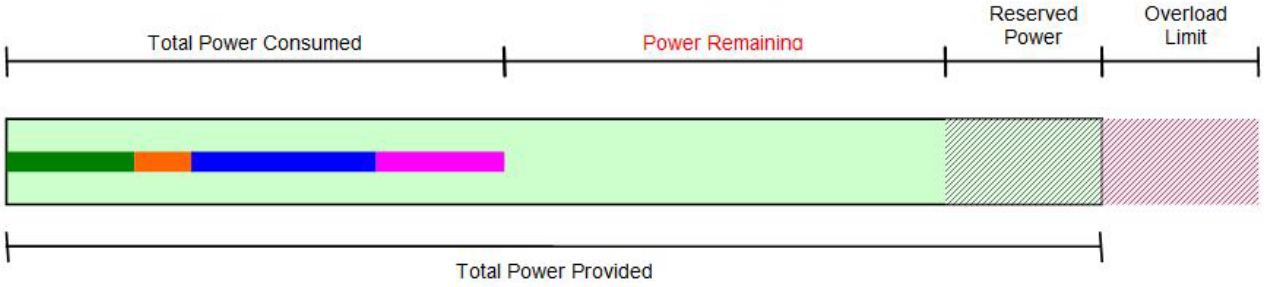



**Figure 5. Consumption-Based Power Management**

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 Si3483 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 Si3483. If a system overload is less than the overload limit, the Si3483 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 Si3483 immediately turns off all low-priority ports. If the system is still overloaded, the Si3483 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 Si3483 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 Si3483 uses the UART interface to communicate with the host MCU. The Si3483 uses the TX and RX pins to send and receive serial data. The BAUD0, BAUD1, and BAUD2 pins select the baud rate for the UART interface.

**Table 5. Baud Rates**

| BAUD2 | BAUD1 | BAUD0 | Baud Rate (bps) |
|---|---|---|---|
| H | L | L | 19200 |
| H | L | H | 38400 |
| H | H | L | 57600 |
| H | 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 Si3483 uses the SPI interface to communicate with the host MCU. The Si3483 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 Si3483 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 Si3483 backs up its configuration to internal flash memory. Once the Si3483 is configured, it is possible to disconnect the host interface and use the Si3483 without a host MCU.

## 3. Serial Packet Protocol

The Si3483 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 Si3483. The Serial Packet Client should be implemented by the user in the host MCU. Skyworks Solutions has reference code available; please contact Skyworks Solutions 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.

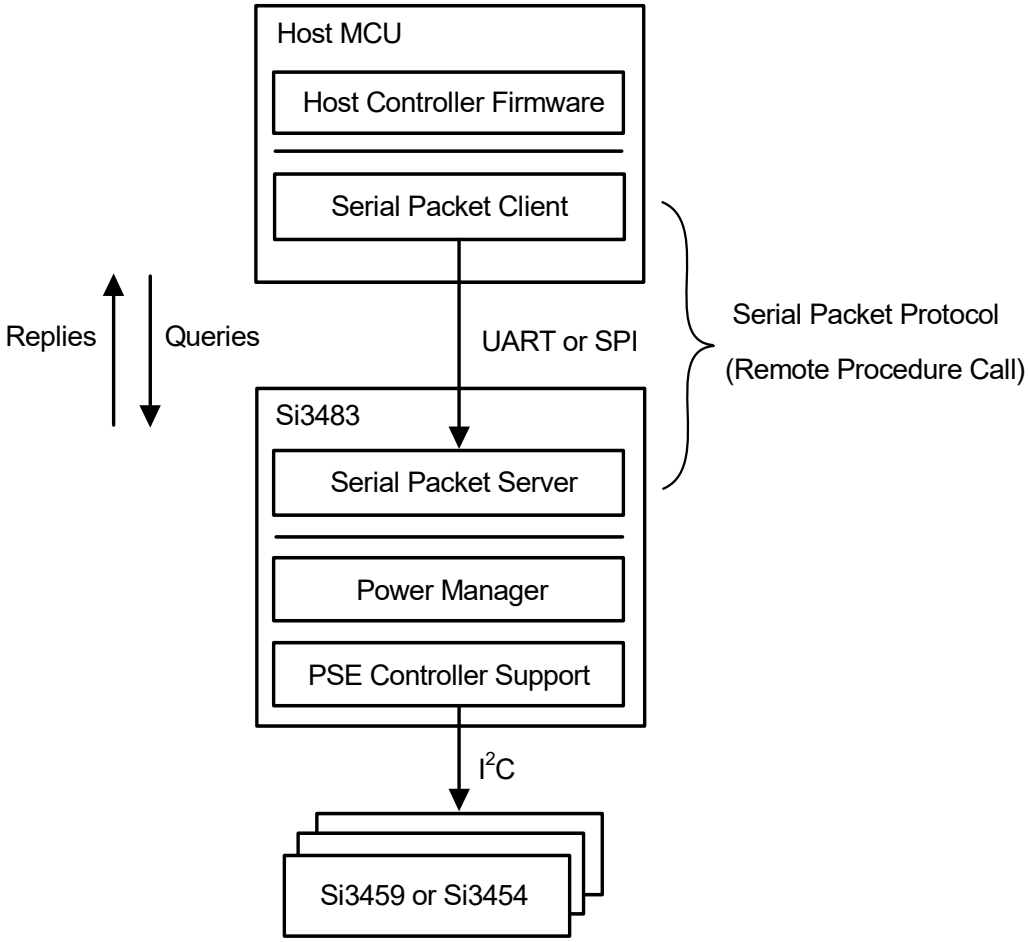



**Figure 6. Serial Packet Protocol**

## 3.1. Packet Format

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

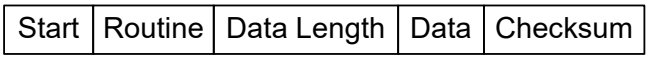



**Figure 7. Packet Format**

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.

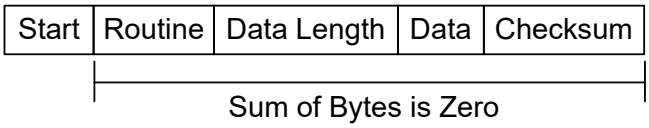



**Figure 8. Packet Checksum**

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.

$$Checksum = (\sim Sum) + 1$$

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 Si3483. 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.

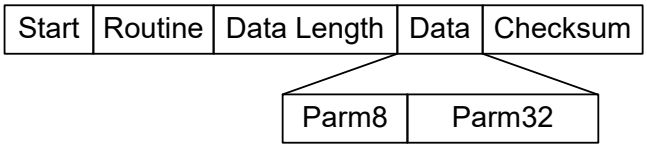



**Figure 9. Parameters Format**

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 | |
| ***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.

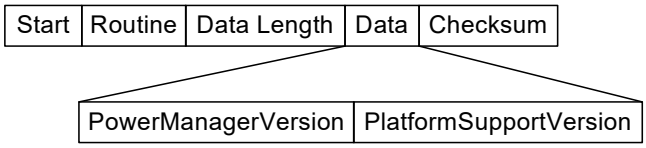



**Figure 10. System Information Format**

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.

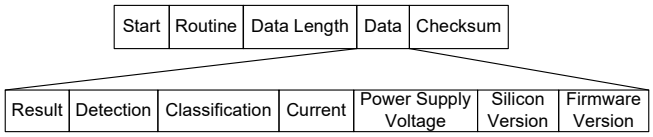



**Figure 11. Port Information Format**

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.

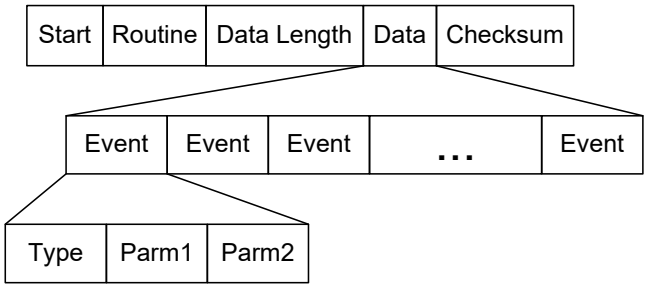



**Figure 12. Events Format**

In the Si3483, 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 Si3483 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 Si3483 checks the configuration every 30 seconds to see if it has changed. If the configuration has changed, the Si3483 backs up the configuration to internal flash memory. While the Si3483 is writing to flash memory, it cannot send or receive packets on the host interface. If a host MCU sends a packet to the Si3483 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 42

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 |

## 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.

## 4.3. System Control

The System Control routines allow a User Interface component to:

- Reset the system
- Restore factory default settings
- Store the configuration immediately in the non-volatile memory

### 4.3.1. Reset System

Reset the system.

**Routine:**

`RTN_RESETSYSTEM`

**Query data:**

Parm8: `None`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Result. Zero (for success) or an error code`

Parm32: `None`

### 4.3.2. Restore Factory Default

Restore the configuration to factory default values.

**Routine:**

`RTN_RESTOREFACTORYDEFAULTS`

**Query data:**

Parm8: `None`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `None`

Parm32: `None`

### 4.3.3. Store Configuration

Store the configuration immediately in the non-volatile memory.

**Routine:**

`RTN_STORECONFIG`

**Query data:**

Parm8: `None`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `None`

Parm32: `None`

## 4.4. Port Control

The Port Control routines allow a User Interface component to:

Set port control

Adjust port power

### 4.4.1. Set Port Control

Controls the method of port turn on and off

**Routine:**

`RTN_SETPORTCONTROL`

**Query data:**

Parm8: `Port number`

Parm32: `Control`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Zero (for success) or an error code`

Parm32: `None`

**Table 13. Control Values**

| Control | Value | Symbol |
|---|---|---|
| Automatic | 0 | PORT_CTRL_AUTOMATIC |
| Force on | 1 | PORT_CTRL_FORCE_ON |
| Force off | 2 | PORT_CTRL_FORCE_OFF |

If the port control is automatic, the Power Manager automatically turns the port on and off when a PD is connected and disconnected from the port. If the port control is forced on, the port's priority is boosted to the forced priority level. This usually results in the port turning on. However, a forced port cannot cause a critical priority port to turn off in order to turn on the forced port. If a forced port is granted power, the Power Manager turns on a forced port even if no PD is detected. If the port control is forced off, the port is unconditionally turned off and held off. A forced-off port is considered to be temporarily off, while a disabled port is considered to be permanently off.

### 4.4.2. Adjust Port Power

Adjust the power granted to a PD.

**Routine:**

`RTN_ADJUSTPORTPOWER`

**Query data:**

Parm8: `Port`

Parm32: `Requested port power in mW`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Zero (for success) or an error code`

Parm32: `None`

An LLDP agent calls this routine to reallocate the power granted to a PD. The agent can request more power than is currently granted or it can request less power than is currently granted. This routine allows an LLDP agent to dynamically change the amount of power granted to a PD during the course of a connection. A port must be on before its power can be adjusted.

## 4.5. System Configuration

The System Configuration routines allow a User Interface component to set and get these items:

- Power provided
- Reserved power
- Overload limit
- Granting policy
- Retry policy
- Power location

### 4.5.1. Set Power Provided

Set the amount of power that is output from a power supply.

**Routine:**

`RTN_SETPOWERPROVIDED`

**Query data:**

Parm8: `Power supply (1 to 3)`

Parm32: `Power provided by the power supply in mW`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Zero (for success) or an error code`

Parm32: `None`

### 4.5.2. Get Power Provided

Get the amount of power that is output from a power supply.

**Routine:**

`RTN_ RTN_GETPOWERPROVIDED`

**Query data:**

Parm8: `Power supply (1 to 3)`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `None`

Parm32: `Power provided by the power supply in mW`

### 4.5.3. Set Reserved Power

Set the percentage of power that is reserved from granting.

**Routine:**

RTN_SETRESERVERPOWER

**Query data:**

Parm8: Reserved power in percentage of the total power provided

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

### 4.5.4. Get Reserved Power

Get the percentage of power that is reserved from granting.

**Routine:**

RTN_GETRESERVEDPOWER

**Query data:**

Parm8: None

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Reserved power in percentage of the total power provided

Parm32: None

If the granting policy is consumption-based, the Power Manager holds this amount of power in reserve. The Power Manager does not use the reserved power to grant new requests. This creates a power buffer that reduces the likelihood of system overloads caused by momentary surges in consumption.

### 4.5.5. Set Overload Limit

Set the maximum system overload that the power supplies can tolerate.

**Routine:**

RTN_SETOVERLOADLIMIT

**Query data:**

Parm8: Overload limit as a percentage of the total power provided

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

### 4.5.6. Get Overload Limit

Get the maximum system overload that the power supplies can tolerate.

**Routine:**

RTN_GETOVERLOADLIMIT

**Query data:**

Parm8: None

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Overload limit as a percentage of the total power provided

Parm32: None

The overload limit is the maximum system overload that the power supplies can tolerate. It is expressed as a percentage of the total power provided. If a system overload is less than the overload limit, the ports are turned off one at a time. If a system overload is greater than the overload limit (severe overload), all of the low-priority ports are immediately turned off.

### 4.5.7. Set Granting Policy

Set the granting policy.

**Routine:**

RTN_SETGRANTINGPOLICY

**Query data:**

Parm8: Granting policy

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

### 4.5.8. Get Granting Policy

Get the granting policy.

**Routine:**

RTN_GETGRANTINGPOLICY

**Query data:**

Parm8: None

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Granting policy

Parm32: None

**Table 14. Granting Policy Values**

| Granting Policy | Value | Symbol |
|---|---|---|
| Grant-based | 0 | GRANT_POLICY_GRANT_BASED |
| Consumption-based | 1 | GRANT_POLICY_CONSUMPTION_BASED |

The granting policy is used by the Power Manager when deciding if a request for power should be granted. If the granting policy is grant based, the remaining power is considered to be the total ungranted power. If the granting policy is consumption-based, the remaining power is considered to be the total unconsumed power (excluding the reserved power). If the remaining power is greater than or equal to the requested power, then the Power Manager grants the request.

Grant based:

```
PowerRemaining = TotalPowerProvided-TotalPowerGranted
```

Consumption based:

```
PowerRemaining = TotalPowerProvided-TotalPowerConsumed-ReservedPower
```

### 4.5.9. Set Retry Policy

Set the retry policy.

**Routine:**

`RTN_SETRETRYPOLICY`

**Query data:**

Parm8: `Retry policy`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Zero (for success) or an error code`

Parm32: `None`

### 4.5.10. Get Retry Policy

Get the retry policy.

**Routine:**

`RTN_GETRETRYPOLICY`

**Query data:**

Parm8: `None`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Retry policy`

Parm32: `None`

**Table 15. Retry Policy Values**

| Retry Policy | Value | Symbol |
|---|---|---|
| Immediate | 0 | RETRY_IMMEDIATELY |
| Reconnect | 1 | RETRY_AFTER_RECONNECT |
| Re-enable | 2 | RETRY_AFTER_REENABLE |

The retry policy specifies when the Power Manager tries again to power a port that is turned off because of a port overload. A port overload is when the power consumed by a PD is greater than the power granted to that PD. If the retry policy is “immediate”, the Power Manager tries to turn the port back on immediately. If the retry policy is “reconnect”, the Power Manager waits until the PD is disconnected and then reconnected before it tries again to power the port. The Power Manager must detect an open circuit on the port before retrying. If the retry policy is “re-enable”, the Power Manager disables the port when a port overload occurs. The user must re-enable the port before the Power Manager tries to power the port again.

## 4.6. Port Configuration

The Port Configuration routines allow a User Interface component to set and get

- Port enable
- Port capability
- Port priority
- Port power limit

### 4.6.1. Set Port Enable

Set whether a port is enabled to turn on.

**Routine:**

`RTN_SETPORTENABLE`

**Query data:**

Parm8: `Port`

Parm32: `Enable (enable: 1, disable: 0)`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Zero (for success) or an error code`

Parm32: `None`

### 4.6.2. Get Port Enable

Get whether a port is allowed to turn on.

**Routine:**

`RTN_GETPORTENABLE`

**Query data:**

Parm8: `Port`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Enable (enabled: 1, disabled: 0) or error code`

Parm32: `None`

### 4.6.3. Set Port Capability

Set whether a port can supply high power.

**Routine:**

RTN_SETPORTCAPABILITY

**Query data:**

Parm8: Port

Parm32: Capability

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

### 4.6.4. Get Port Capability

Get whether a port is allowed to turn on.

**Routine:**

RTN_GETPORTCAPABILITY

**Query data:**

Parm8: Port

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Capability or error code

Parm32: None

**Table 16. Port Capability Values**

| Capability | Value | Symbol |
|---|---|---|
| Low power | 0 | CAPABILITY_LOW_POWER |
| High power | 1 | CAPABILITY_HIGH_POWER |

If the port hardware is designed to supply high power (PoE+), set Capability to one. Otherwise, set Capability to zero. Note that a port's capability cannot be changed while the port is on

### 4.6.5. Set Port Power Location

Set the location of the power source.

**Routine:**

RTN_SETPORTPOWERLOCATION

**Query data:**

Parm8: Port

Parm32: Location

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

### 4.6.6. Get Port Power Location

Get the location of the power source.

**Routine:**

RTN_GETPORTPOWERLOCATION

**Query data:**

Parm8: Port

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Location or error code

Parm32: None

**Table 17. Power Location Values**

| Location | Value | Symbol |
|---|---|---|
| Endpoint | 0 | LOCATION_ENDPOINT |
| Midspan | 1 | LOCATION_MIDSPAN |

If the power source is within an Ethernet switch, the location is “endpoint”. If the power source is inserted between an Ethernet switch and a PD, the location is “midspan”. The Power Manager uses different back-off timings for different locations. The Power Manager assumes that an endpoint device uses the Alternative A pinout and that a midspan device uses the Alternative B pinout. For alternative A, power is applied to wire pairs 1,2 and 3,6. For alternative B, power is applied to wire pairs 4,5 and 7,8

(the spare pairs in the case of 10/100 Ethernet). Conventionally, alternative B is used for midspan power injectors. For alternative B, detection is done with over 2 seconds between detection pulses so as to avoid interfering with end-point equipment trying to provide power using alternative A.

### 4.6.7. Set Port Priority

Set the priority of a port.

**Routine:**

`RTN_SETPORTPRIORITY`

**Query data:**

Parm8: `Port`

Parm32: `Priority`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Zero (for success) or an error code`

Parm32: `None`

### 4.6.8. Get Port Priority

Get the priority of a port.

**Routine:**

`RTN_GETPORTPRIORITY`

**Query data:**

Parm8: `Port`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Priority or error code`

Parm32: `None`

**Table 18. Port Priority Values**

| Priority | Value | Symbol |
|---|---|---|
| Low | 0 | PRIORITY_LOW |
| High | 1 | PRIORITY_HIGH |
| Critical | 3 | PRIORITY_CRITICAL |

The priority of a port indicates how important it is that the port receives power. If there is not enough power provided for all ports that want power, then the low priority ports are the first ports to be denied. Critical priority ports are the last ports to be denied.

If a port is forced on, then the port's priority is elevated to the forced priority level. Forced priority is between high priority and critical priority and cannot be directly set by the user. When a port is forced on, it may cause a high priority port to be turned off, but it can never cause a critical priority port to be turned off.

If a severe overload occurs, all of the low priority ports are immediately powered off.

### 4.6.9. Set Port Legacy Support

Enable or disable legacy support.

**Routine:**

RTN_SETPORTLEGACYSUPPORT

**Query data:**

Parm8: Port

Parm32: Enable (enable: 1, disable: 0)

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

If disabled then only IEEE standard 802.3AF- or AT-compatible PDs will be detected. Otherwise non-standard PDs with large common-mode capacitance (legacy PDs) will be detected as well.

### 4.6.10. Get Port Legacy Support

Get the legacy support setting of a port

**Routine:**

RTN_GETPORTLEGACYSUPPORT

**Query data:**

Parm8: Port

Parm32: None

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Enable (enable: 1, disable: 0) or error code

Parm32: None

If enabled the legacy PDs will be detected.

### 4.6.11. Set Port Power Limit

Set the power limit of a port.

**Routine:**

RTN_SETPORTPOWERLIMIT

**Query data:**

Parm8: Port

Parm32: Limit, maximum power that may be granted to a port in mW

**Reply packet format:**

Parameters

**Reply data:**

Parm8: Zero (for success) or an error code

Parm32: None

### 4.6.12. Get Port Power Limit

Get the power limit of a port

**Routine:**

`RTN_GETPORTPOWERLIMIT`

**Query data:**

Parm8: `Port`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `None`

Parm32: `Limit, maximum power that may be granted to a port in mW`

Power limit restricts the amount of power that may be granted to a port. If a port's power limit is zero, the Power Manager grants the power requested without restriction. If a port's power limit is greater than zero, the Power Manager grants the lesser of the power limit or the power requested. If a power request is greater than the power limit, the Power Manager grants less power than requested.

## 4.7. Power Supply Status

The Power Supply Status routine allows a User Interface component to get:

Power Supply Status

### 4.7.1. Get Power Supply Status

Get the status of a power supply.

**Routine:**

`RTN_GETPOWERSUPPLYSTATUS`

**Query data:**

Parm8: `Port`

Parm32: `None`

**Reply packet format:**

`Parameters`

**Reply data:**

Parm8: `Status`

Parm32: `None`

**Table 19. Power Supply Status Values**

| Status | Value | Symbol |
|---|---|---|
| Removed | 0 | STATUS_POWER_SUPPLY_REMOVED |
| Inserted | 1 | STATUS_POWER_SUPPLY_INSERTED |

A User Interface component calls this function to determine whether a power supply is present in a bay.

If the voltage on the specified power supply pin (PS1, PS2, or PS3) is high, this routine returns STATUS_POWER_SUPPLY_INSERTED, otherwise, this routine returns STATUS_POWER_SUPPLY_REMOVED.

## 4.8. Events

The Event routine allows a User Interface component to get:

- System events
- Port events
- Power supply events
- Error events
- Informational events

### 4.8.1. Get Events

Get the events from the event queue.

**Routine:**

`RTN_GETEVENTS`

**Query data:**

Parm8: `None`

Parm32: `None`

**Reply packet format:**

`Events`

**Reply data:**

This routine returns a series of event descriptors (maximum 72) which consist of 3 bytes each:

Byte 0: `Event type`

Byte 1: `Parm1, value depends on event type`

Byte 2: `Parm2, optional, value depends on event type`

**Table 20. Event Type Values**

| Event Type | Value | Symbol |
|---|---|---|
| System | 1 | EVENT_TYPE_SYTSEM |
| Port | 2 | EVENT_TYPE_PORT |
| Power supply | 4 | EVENT_TYPE_POWER_SUPPLY |
| Error | 8 | EVENT_TYPE_ERROR |
| Information | 16 | EVENT_TYPE_INFO |

Parameters Parm1 and Parm2 depend on event type:

- System event:
  - Parm1: value corresponding to System status values (see Table 9.)
  - Parm2: None
- Port event:
  - Parm1: value corresponding to Port status values (see Table 10.)
  - Parm2: port number
- Power supply event:
  - Parm1: value corresponding to Power supply status values (see Table 20.)
  - Parm2: power supply number
- Error event:
  - Parm1: value corresponding to System status values (see Table 23.)
  - Parm2: error specific

- Informational event:
  - Parm1: value corresponding to System status values (see Table 22.)
  - Parm2: None

**Table 21. Information Code Values**

| Event Type | Value | Symbol |
|---|---|---|
| Restored to factory defaults | 1 | INFO_DEFAULTS_RESTORED |
| System reset | 2 | INFO_SYSTEM_RESET |
| Configuration saved | 3 | INFO_CONFIG_SAVED |

## 4.9. Return Codes

The routines of the Power Manager API return codes to indicate the success or failure of an operation. These codes are also used in Parm1 of error events and information events.

**Table 22. Return Code Values**

| Return Code | Value | Symbol |
|---|---|---|
| Success | 0 | SUCCESS |
| Port number is invalid | –1 | ERROR_PORT_INVALID |
| Power supply number is invalid | –2 | ERROR_PWR_SUPLY_INVALID |
| Parameter is invalid | –3 | ERROR_PARAMETER_INVALID |
| Cannot create resource | –4 | ERROR_RESOURCE_CREATE |
| Resource is invalid | –5 | ERROR_RESOURCE_INVALID |
| Cannot configure resource | –6 | ERROR_RESOURCE_CONFIG |
| Cannot read from resource | –7 | ERROR_RESOURCE_READ |
| Cannot write to resource | –8 | ERROR_RESOURCE_WRITE |
| Cannot find the resource | –9 | ERROR_RESOURCE_NOT_FND |
| Cannot load the configuration | –10 | ERROR_CONFIG_LOAD |
| Cannot save the configuration | –11 | ERROR_CONFIG_SAVE |
| Configuration data is invalid | –12 | ERROR_CONFIG_INVALID |
| Configuration data is corrupt | –13 | ERROR_CONFIG_CORRUPT |
| System overload | –14 | ERROR_SYSTEM_OVERLOAD |
| Port overload | –15 | ERROR_PORT_OVERLOAD |
| Startup overload | –16 | ERROR_STARTUP_OVERLOAD |

## 5. Pin Descriptions

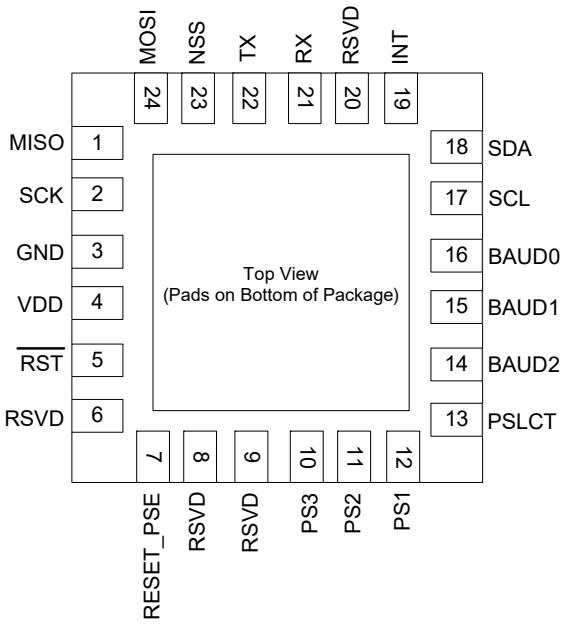



**Table 23. Si3483 Pin Descriptions**

| Pin # | Name | Type | Description |
|---|---|---|---|
| 1 | MISO | Output | SPI output. |
| 2 | SCK | Input | SPI clock. |
| 3 | GND | Power | Ground. |
| 4 | VDD | Power | VDD. |
| 5 | RST | Input | Reset (a low will reset the Si3483). |
| 6 | RSVD | Input | Reserved—tie low. |
| 7 | RESET_PSE | Output | Reset output for connection to reset input pins of Si3459 or Si3454 PSE controllers. |
| 8 | RSVD | Reserved | Do not connect. |
| 9 | RSVD | Reserved | Do not connect. |
| 10 | PS3 | Input | Logic high indicates the power supply is available. |
| 11 | PS2 | Input | Logic high indicates the power supply is available. |
| 12 | PS1 | Input | Logic high indicates the power supply is available. |
| 13 | PSLCT | Input | Tie high or low to select between SPI and UART interface. |
| 14 | BAUD2 | Input | Tie high or low to select UART baud rate. |
| 15 | BAUD1 | Input | Tie high or low to select UART baud rate. |

**Table 23. Si3483 Pin Descriptions (Continued)**

| Pin # | Name | Type | Description |
|---|---|---|---|
| 16 | BAUD0 | Input | Tie high or low to select UART baud rate. |
| 17 | SCL | Open Collector | Connect to Si3459 SCL and pull up resistor. |
| 18 | SDA | Open Collector | Connect to Si3459 SDA and pull up resistor. |
| 19 | INT | Input | Connect to Si3459 INT and pull up resistor. |
| 20 | RSVD | Reserved | Do not connect. |
| 21 | RX | Input | UART receive. |
| 22 | TX | Output | UART transmit. |
| 23 | NSS | Input | SPI select. |
| 24 | MOSI | Input | SPI input. |

# 6. Ordering Guide

**Table 24. Si3483 Ordering Guide**

| Ordering Part Number | Description | Package Information |
|---|---|---|
| Si3483-A02-GM | Power management controller | 24-pin 4x4 mm QFN<br>RoHS compliant |
| Si3459SMART24-KIT | An evaluation kit and PSE reference design with the Si3483, three Si3459 PSE Port controllers, digital bus isolation, and configuration and debug features. | Evaluation Kit |

**Notes:**
1. Add “R” to the part number to denote tape and reel option (Si3483-A02-GMR).
2. The ordering part number is not the same as the device mark. See "7. Package Outline: 24-Pin QFN" on page 46 for device marking information.

## 7. Package Outline: 24-Pin QFN

The Si3483 is packaged in an industry-standard, RoHS-compliant 4x4 mm$^2$, 24-pin QFN package.

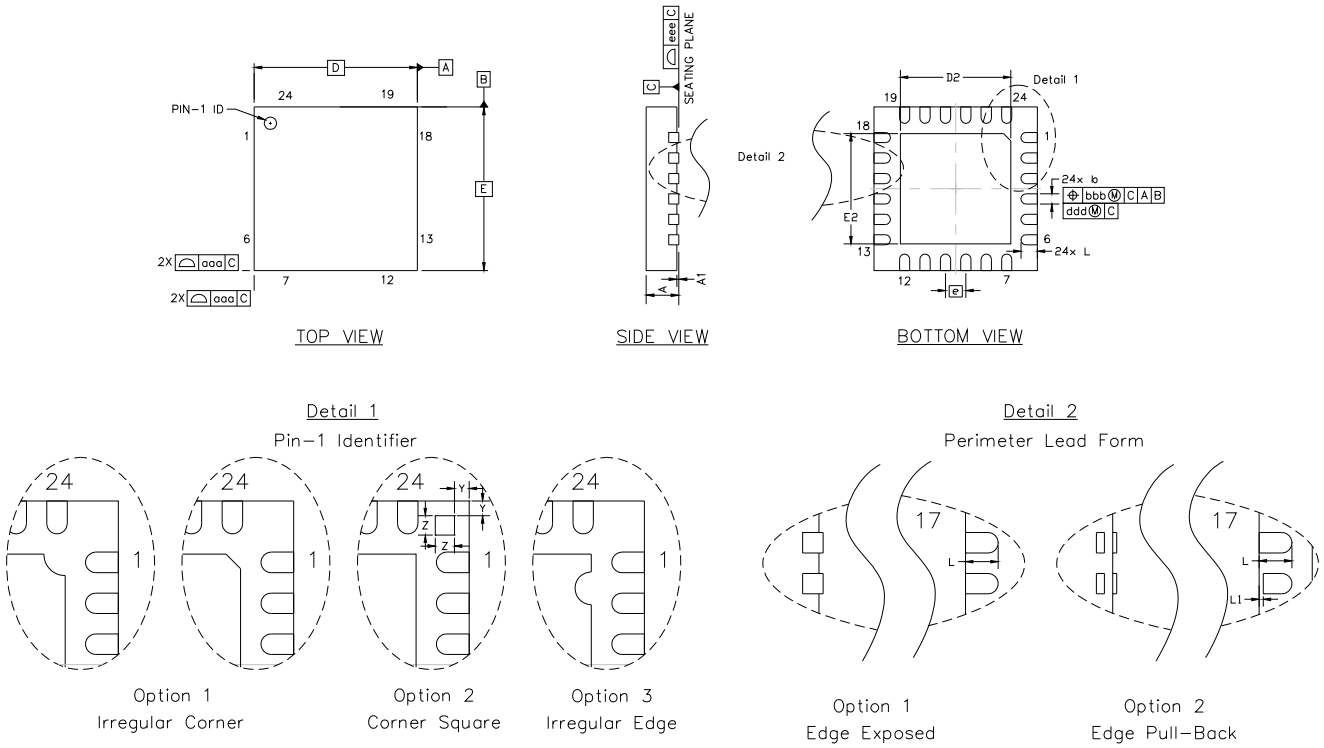



**Figure 13. 24-Pin QFN Mechanical Diagram**

**Table 25. QFN-24 Package Dimensions**

| Dimension | Min | Nom | Max |
|---|---|---|---|
| A | 0.70 | 0.75 | 0.80 |
| A1 | 0.00 | 0.02 | 0.05 |
| b | 0.18 | 0.25 | 0.30 |
| D | | 4.00 BSC | |
| D2 | 2.55 | 2.70 | 2.80 |
| e | | 0.50 BSC | |
| E | | 4.00 BSC | |
| E2 | 2.55 | 2.70 | 2.80 |
| L | 0.30 | 0.40 | 0.50 |
| L1 | 0.00 | — | 0.15 |
| aaa | — | — | 0.15 |
| bbb | — | — | 0.10 |
| ddd | — | — | 0.05 |
| eee | — | — | 0.08 |
| Z | — | 0.24 | — |
| Y | — | 0.18 | — |

**Notes:**
1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to the JEDEC Solid State Outline MO-220, variation WGGD except for custom features D2, E2, Z, Y, and L which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

## 8. PCB Land Pattern

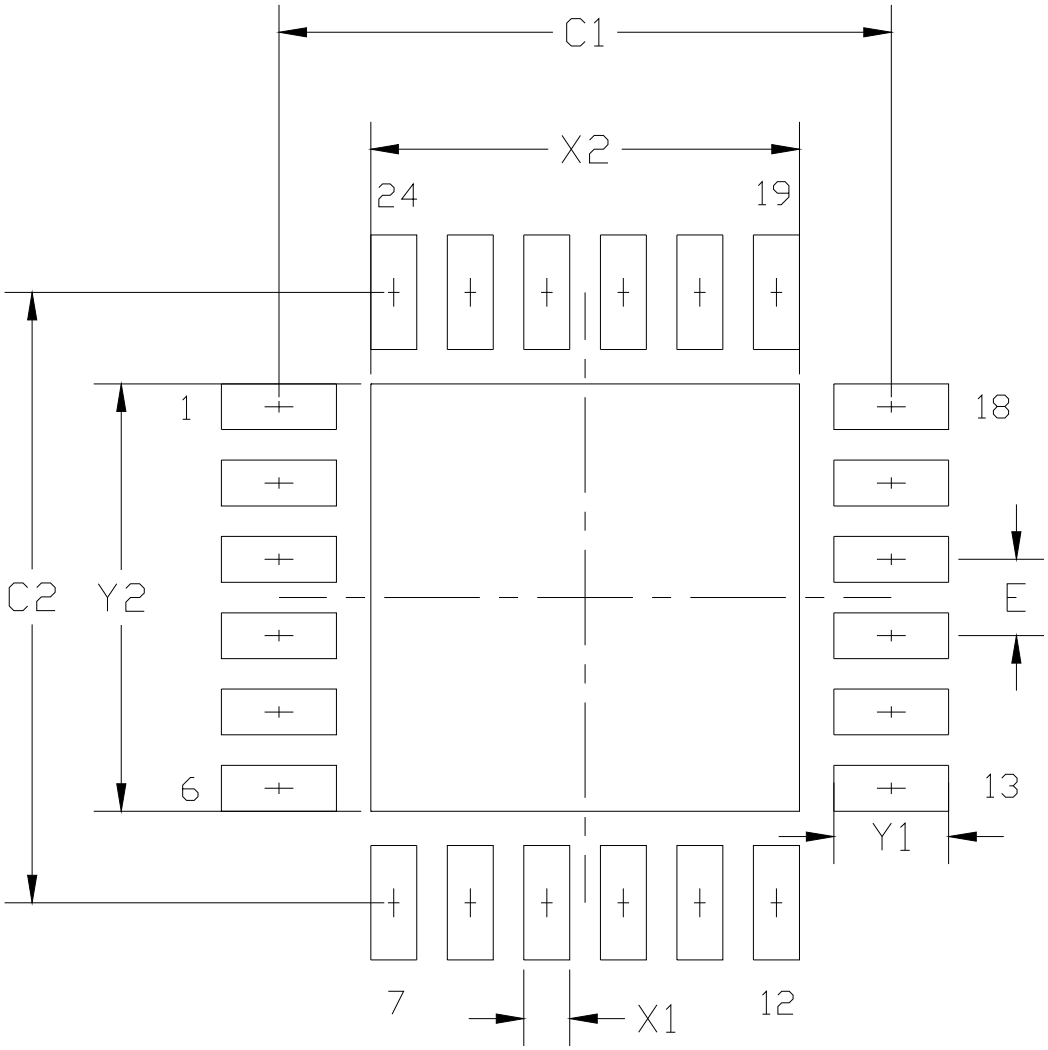



**Figure 14. Typical QFN-24 PCB Land Pattern**

**Table 26. QFN-24 PCB Land Pattern Dimensions**

| Dimension | MIN | MAX |
|---|---|---|
| C1 | 3.90 | 4.00 |
| C2 | 3.90 | 4.00 |
| E | 0.50 BSC | |
| X1 | 0.20 | 0.30 |
| X2 | 2.70 | 2.80 |
| Y1 | 0.65 | 0.75 |
| Y2 | 2.70 | 2.80 |

**Notes:**

**General**

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. This land pattern design is based on the IPC-7351 guidelines.

**Solder Mask Design**

3. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60mm minimum, all the way around the pad.

**Stencil Design**

4. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
5. The stencil thickness should be 0.125mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.
7. A 2x2 array of 1.10mm x 1.10mm openings on 1.30mm pitch should be used for the center ground pad.

**Card Assembly**

8. A No-Clean, Type-3 solder paste is recommended.
9. The recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

## 9. Top Marking Diagram

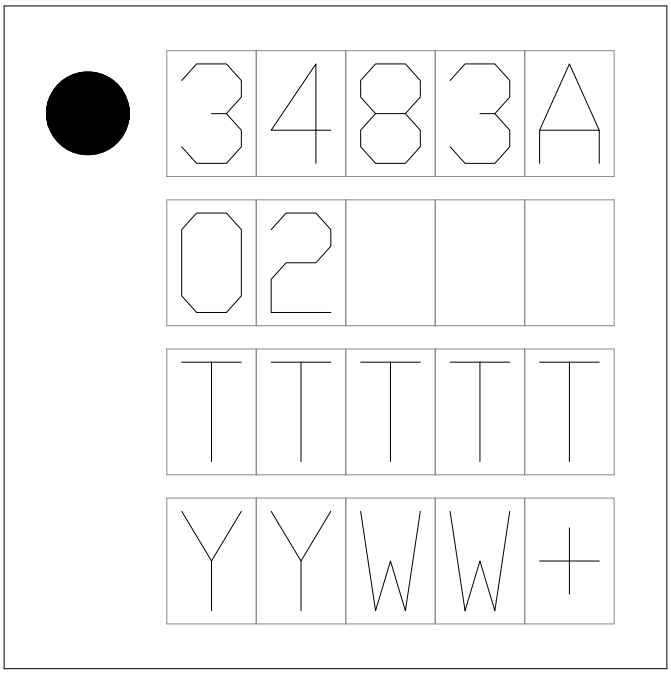



**Figure 15. Top Marking Diagram**

**Table 27. Top Marking Explanation**

| | | |
|---|---|---|
| Line 1 Marking: | Pin 1 Identifier | Circle, 0.5 mm diameter |
| | Product ID | 3483A |
| Line 2 Marking: | Firmware revision | 02 = Firmware revision 02 |
| Line 3 Marking: | TTTTT = Trace Code | Manufacturing code characters from the Markings section of the Assembly Purchase Order form |
| Line 4 Marking: | YYWW+ = Date Code | YY = Last two digits of current year<br>WW = Current Work Week |
| | Lead Free Designator | + |

# Document Change List

## Revision 1.0 to Revision 1.1

- Reformatted commands in "4. Power Manager API" on page 15.

SKYWORKS®

**Connecting Everyone and Everything, All the Time**



**Portfolio**
www.skyworksinc.com

**Quality**
www.skyworksinc.com/quality

**Support & Resources**
www.skyworksinc.com/support

**Copyright © 2021 Skyworks Solutions, Inc. All Rights Reserved.**

Information in this document is provided in connection with Skyworks Solutions, Inc. ("Skyworks") products or services. These materials, including the information contained herein, are provided by Skyworks as a service to its customers and may be used for informational purposes only by the customer. Skyworks assumes no responsibility for errors or omissions in these materials or the information contained herein. Skyworks may change its documentation, products, services, specifications or product descriptions at any time, without notice. Skyworks makes no commitment to update the materials or information and shall have no responsibility whatsoever for conflicts, incompatibilities, or other difficulties arising from any future changes.

No license, whether express, implied, by estoppel or otherwise, is granted to any intellectual property rights by this document. Skyworks assumes no liability for any materials, products or information provided hereunder, including the sale, distribution, reproduction or use of Skyworks products, information or materials, except as may be provided in Skyworks' Terms and Conditions of Sale.

THE MATERIALS, PRODUCTS AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING FITNESS FOR A PARTICULAR PURPOSE OR USE, MERCHANTABILITY, PERFORMANCE, QUALITY OR NON-INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT; ALL SUCH WARRANTIES ARE HEREBY EXPRESSLY DISCLAIMED. SKYWORKS DOES NOT WARRANT THE ACCURACY OR COMPLETENESS OF THE INFORMATION, TEXT, GRAPHICS OR OTHER ITEMS CONTAINED WITHIN THESE MATERIALS. SKYWORKS SHALL NOT BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO ANY SPECIAL, INDIRECT, INCIDENTAL, STATUTORY, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUES OR LOST PROFITS THAT MAY RESULT FROM THE USE OF THE MATERIALS OR INFORMATION, WHETHER OR NOT THE RECIPIENT OF MATERIALS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Skyworks products are not intended for use in medical, lifesaving or life-sustaining applications, or other equipment in which the failure of the Skyworks products could lead to personal injury, death, physical or environmental damage. Skyworks customers using or selling Skyworks products for use in such applications do so at their own risk and agree to fully indemnify Skyworks for any damages resulting from such improper use or sale.

Customers are responsible for their products and applications using Skyworks products, which may deviate from published specifications as a result of design defects, errors, or operation of products outside of published parameters or design specifications. Customers should include design and operating safeguards to minimize these and other risks. Skyworks assumes no liability for applications assistance, customer product design, or damage to any equipment resulting from the use of Skyworks products outside of Skyworks' published specifications or parameters.

Skyworks, the Skyworks symbol, Sky5®, SkyOne®, SkyBlue™, Skyworks Green™, Clockbuilder®, DSPLL®, ISOmodem®, ProSLIC®, and SiPHY® are trademarks or registered trademarks of Skyworks Solutions, Inc. or its subsidiaries in the United States and other countries. Third-party brands and names are for identification purposes only and are the property of their respective owners. Additional information, including relevant terms and conditions, posted at www.skyworksinc.com, are incorporated by reference.

Skyworks Solutions, Inc. | Nasdaq: SWKS | sales@skyworksinc.com | www.skyworksinc.com
USA: 781-376-3000 | Asia: 886-2-2735 0399 | Europe: 33 (0)1 43548540 |