TPS2384 PoE Firmware Version 2.4

User's Guide

Literature Number: SLVU195
June 2007

Texas Instruments

Contents

List of Figures

List of Tables

Read This First

This manual contains information that enables customers to use firmware for the MSP430 that supports the TPS2384 Power Controller device from Texas Instruments.

This guide is a functional and design description of the firmware, and describes its use on the TPS2384 reference system.

Use of this firmware and GUI is bound by the included Software License Agreement located in this folder.

Related Documentation from Texas Instruments

Documents related to Power over Ethernet firmware for the TPS2384 are:

1. Quad Ethernet Power Sourcing Equipment Power Manager (TPS2384 PSE data sheet) (SLUS634)
2. TPS2384 User's Guide (User's Guide for the TPS2384 Reference Design) (SLVU126)
3. *PSE Controller Interface Control* (Design guidelines for MSP430 controller for use with TPS23284 PSE) (SLUA328)
4. *Firmware release 2.4 for the TPS2384 PSE (Release Notes)*
5. *Power Over Etherner Power Sourcing Equipment Software License Agreement*

If You Need Assistance

For assistance with Power over Ethernet firmware and related issues, contact your local Texas Instruments sales representative.

TPS2384 PoE Firmware Version 2.4

1.1 Overview

The TPS2384 PSE is a 4-channel Power-over-Ethernet (PoE) Power Sourcing Equipment (PSE) manager. The TPS2384 manages power distribution to powered devices (PDs) over Ethernet cabling according to the IEEE 802.3af specification. The TPS2384 can be used in one of two modes: *auto mode* or *port management (manual) mode*.

In auto mode, the TPS2384 provides basic IEEE 802.3af-compliant power to PDs over a RJ-45 Ethernet physical interface. It automatically detects, classifies and powers ports, according to this standard. The TPS2384 is a standalone device in auto mode, and does not require any type of controller.

Port-management or manual mode allows more control over the PSE so that additional functionality can be provided by the PoE subsystem. In manual mode, the TPS2384 works in conjunction with a MSP430 microcontroller that provides all the instructions to the TPS2384 for the various states of device discovery and powering. Manual-mode functionality such as Power Management, Legacy Device Detection and other features are described later in this document. The remainder of this document describes the manual mode operation of the PoE system with the MSP430 controller.

One MSP430 microcontroller can control up to 12 TPS2384s simultaneously, for a total of 48 ports in a system. The diagram below shows a simplified block diagram for a 24-port system with a MSP430 microcontroller. The MSP430 interfaces to the TPS2384 devices through a standard I^2C bus, where the MSP430 is the bus master. The MSP430 interfaces to the host controller either through RS-232 or I^2C.

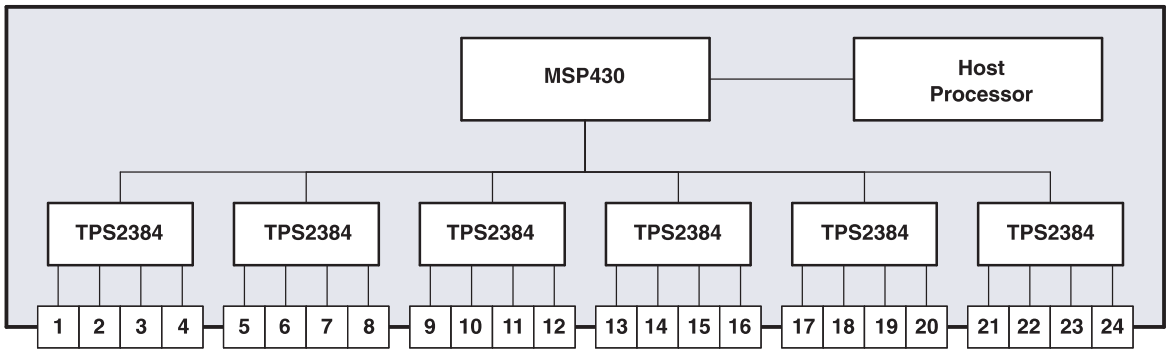

Figure 1-1. PoE System Block Diagram (24-port system)

The MSP430 lets system designers manage many PoE controllers in one system with cost, power, and space constraints. In a switch, it offloads the low-level device specific tasks and power management responsibilities from the CPU.

For more information on Texas Instruments TPS2384 PoE controller and associated firmware, see the product pages at:

http://focus.ti.com/docs/prod/folders/print/tps2384.html

This User's Guide describes the content, design, and use of this firmware.

1.1.1 Firmware Features and Support

The MSP430 firmware supports the TPS2384 PSE in port-management mode, providing a number of features in a PoE system.

The firmware supports the following system configurations and devices:

- TPS2384 PSE controller, revision G
- Firmware runs on the MSP430F148 or MSP430F169 microcontroller on the TPS2384 Reference Design
- The firmware supports modules and discreet solutions with the TPS2384 that include a port capacitor between the P and N pins with a value of 220 nF. If the module or discreet solution contains a port capacitor of a different value, the firmware will not detect devices reliably and will be non-compliant with IEEE 802.3al.
- The firmware supports any number of ports between 4 and 48 that are a multiple of 4. (1–12 TPS2384 4-channel devices)
- The firmware supports PoE modules from a number of vendors that contain the TPS2384 4-channel PSE.
- The system supports either an RS-232 or I^2C inteface between the MSP430 and the host. If a RS-232 interface is used, then a MSP430F148 or MSP430F169 controller will work. If an I^2C interface is used, then the MSP430F169 controller must be used.
- The reference design also comes with a **GUI** that can be used for configuration and control of the application on the reference design.

The firmware supports the following features:

- √ Fully compliant with IEEE802.3af Power-over-Ethernet specification
- √ Device Discovery and Classification
- √ AC or DC disconnect
- √ Power Management
- √ Legacy Device Detection
- √ Logical-to-Physical Port Mapping
- √ Save-to-Flash and Restore-Factory-Defaults
- √ Firmware Download Support

1.1.2 Passive Component Values

The TPS2384 device has several passive components that can be used with it that affect the system in different ways. The software was designed to work with the specific component values shown in Table 1-1. If a system is designed using a different value for the given components, then the software may not work perfectly and may need to be modified for the specific application.

Table 1-1. Reference Design Passive Component Values

Device	Location	Purpose	Value Supported by Unmodified Firmware
Port Capacitor	Between the P and N pins on each TPS2384 port	Controls the power ramp-up and ramp-down rates. (According to the IEEE 802.3af specification, a wide range of port capacitance values may be used.)	220 nF
AC Disconnect Capacitor	There is a capacitor between the AC high and AC low signals (on the other side of a switch).	Controls the characteristics of the AC Disconnect waveform (frequency, etc.)	68 µF/100V

See the data sheet on the TPS2384 and the PoE controller interface control document in the *Related Information* section above for more information.

1.1.2.1 Port Capacitor

There is a port capacitor between the P and N pins on each port of the TPS2384. This controls the rate at which power ramp up and down happens. According to the IEEE 802.3af specification, there is a wide range of port capacitance that is allowed to be used here.

The firmware for the MSP430 that supports the TPS2384 is written to assume a value of 220 nF for this port capacitor. Other values for the port capacitor will require customization of the firmware.

1.1.2.2 AC Disconnect Capacitor

There is a capacitor between the AC high and AC low signals (on the other side of a switch). This capacitor controls the characteristics of the AC Disconnect waveform (frequency, etc.) and the software relies on a value of 68 F/100V device.

1.1.2.3 Serial Interface

The host serial interface uses the MSP430's on board UART. It is run at 19.2K baud set to 8 data bits, no parity and 1 stop bit.

1.2 Firmware Design

This firmware runs on the MSP430 to control the TPS2384, and supports systems with up to 48 ports; up to 12 TPS2384 devices, each with 4 ports. The firmware controls the system-level parameters as well as port-level parameters for each TPS2384 device within the system.

The firmware uses a state machine approach to accomplish this. The state machine tracks the operational state of each port on any given TPS2384. The firmware iterates through the TPS devices and through the ports, servicing ports in a fixed sequence.

There are two interrupts that can occur any time; one for serial communications (either RS-232 or I^2C) with the host, while the other interrupt is for a fixed system-timer tick that controls the state machine timing.

The firmware also keeps track of system and device error conditions that occur, as well as any administrative events that affect the port states, such as administratively disabling a port, or changing a port configuration to not support a particular feature.

The main loop of the firmware that iterates through these states is in the *main.c* file. For DC-disconnect mode, each main loop takes approximately 75 milliseconds. For AC-disconnect mode, each main loop takes approximately 25 milliseconds. A high-level diagram of the states implemented in the firmware is shown in Figure 1-2.

Note that not every state is shown in Figure 1-2. States are lumped together in groups for clarity. Currently there are more than 30 different states.

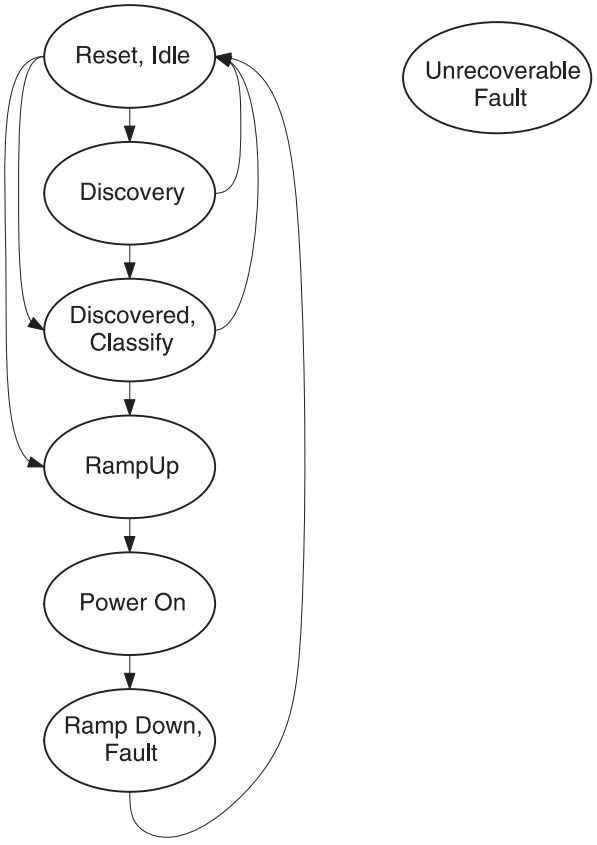

Figure 1-2. Firmware Port State Machine Overview

Descriptions of each state are given below.

1.2.1 Reset, Idle

The *Reset/Idle* state group has only two states –*idle* and *reset_port*.

Idle is the initial state of the system at startup. This state can be entered from a number of places, depending on system or device changes that occur as the machine is advancing through the states. If the port and discovery are enabled, idle advances to the *pre_discovery* state in the *Discovery* group.

- If the IEEE specification test mode is selected, it must be powered up immediately. In this case, *idle* goes either to *discovered_test* in the *Discovered, Classify* group, or to *ramp_up_power*. It only goes to *ramp_up_power* if power management has already enabled power to this port. Normally, it goes to *discovered_test* and sets the discovered bit to tell the power management task that it needs power. Even though test is intended to power up the port regardless of port state, it is still subject to power management.
- The *Reset_port* state sends a command to the TPS2384 to reset the port, then always goes to the *idle* state. It is provided for states where there is no time left in the time slice to send a reset command on the I^2C port.

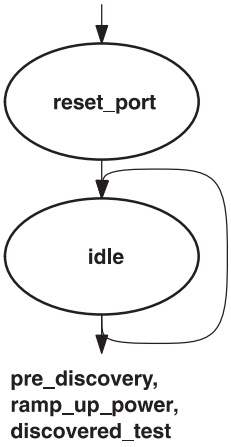

Figure 1-3. Reset, Idle States Details

1.2.2 Discovery

The *Discovery* states support the process of device discovery for IEEE 802.3af-compliant PDs, and for Legacy PDs. These states are always entered from the idle state.

If a valid PD is discovered, the state machine proceeds to either the discovered/classify set of states, or directly to the ramp up state. If no PD is found, the discovery states periodically cycle back to the idle state to start over again.

Figure 1-4 shows the state transitions, and the text following that describes what happens in each specific state.

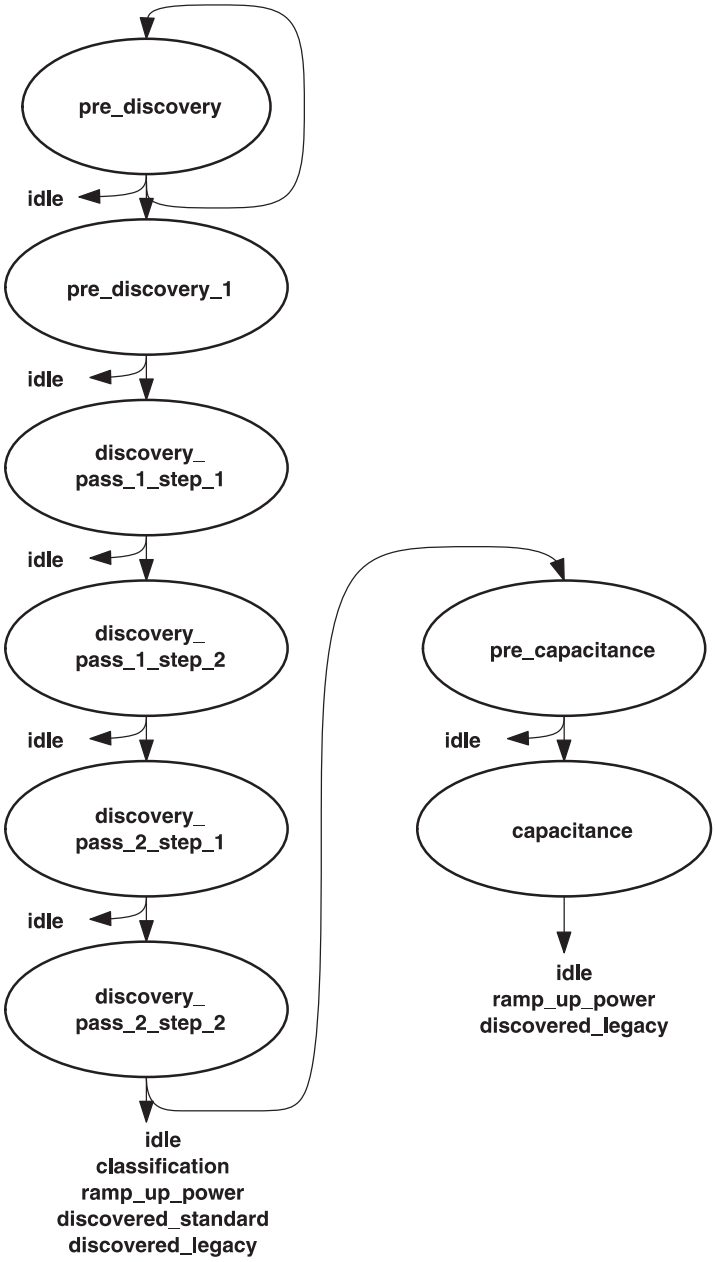

Figure 1-4. Reset, Idle States Details

1.2.2.1 Pre_discovery State

The TPS2384 has leakage current when measurements are being made. The leakage current pulls up the voltage on the port when there is no device present to consume the currrent and pull the voltage down. If the voltage is above a threshold, two things are true:

1. There is nothing plugged into the port
2. It is not possible to get the voltage correct for discovery.

The *pre_discovery* state measures the voltage on the port. If the voltage goes below a specific threshold, the state is changed to *pre_discovery_1*. If the voltage does not go below the threshold in a specified number of port cycles, the port is reset and sent back to the *idle* state to start over again.

1.2.2.2 Pre_discovery_1 State

The *pre_discovery_1* state reads the voltage on the port a second time to verify that it is still below the threshold required for discovery. If it is, it starts the 4.4 volt measurement of current and advances the state to *discovery_pass_1_step_1*. If it is not, it resets the port and sends the state back to *idle*.

1.2.2.3 Discovery states

The Standard Discovery process on the TPS2384 firmware uses a 4-meaurement-point discovery. Instead of simply applying 4.4 and 8.8 volts to the port and measuring the currents, it goes through the sequence twice. This enables it to filter out capacitive loads that might look like resistive loads with only 2 meaurement points. It also enables it to identify certain types of legacy PDs that have some capacitance in their signatures.

All of the discovery states first check to ensure that the port is still enabled. If it is not, they reset the port and set it back to the *idle* state. If it is, they finish a measurement, generally start a measurement, and go on to the next state in the sequence.

Discovery_pass_1_step_1 State	Reads and stores the current for the 4.4 volt test started by pre_discovery_1. It starts an 8.8 volt current measurement.
Discovery_pass_1_step_2 State	Reads and stores the current for the 8.8 volt test started by Discovery_pass_1_step_1. It starts a second 4.4 volt current measurement.
Discovery_pass_2_step_1 State	Reads and stores the current for the 4.4 volt test started by Discovery_pass_1_step_2. It starts a second 8.8 volt current measurement.
Discovery_pass_2_step_2 State	Reads and stores the current for the 8.8 volt test started by *Discovery_pass_2_step_1*. It then does up to three discovery processes if they are enabled.

If standard discovery is enabled, it uses 4 current differences, one for each pair of 4.4 and 8.8 volt currents. If all 4 are within an acceptable range, the PD is marked as discovered. If powerup is enabled and no other port on the TPS2384 is operating the output FET in the linear region, classification is enabled and the state is set to *classification*. Otherwise, the state is set to *discovered_standard*.

If legacy discovery is enabled, and no standard discovery occurs, the firmware compares the 4 currents to the several sets of windows for current derived from various legacy devices. Legacy devices are devices which do not meet the IEEE standard for PoE, but can nonetheless be powered up. As mentioned before, the 4 point discovery method makes it possible to identify the resistive and capacitive signatures of these devices simply by measuring the 4 currents.

If a legacy device is discovered, and powerup is enabled for this port, ramp up is started, and the state is set to *ramp_up_power*. This is done only if no other FET on the TPS2384 is in the linear mode. Ramp up and classification operate the power FET on the port in the linear mode, which increases the power dissipation of the TPS2384. For this reason, only one port at a time is allowed to do this.

If powerup is not enabled, the state is set to *discovered_legacy*.

The third discovery mode is capacitive discovery. If no other discovery occurs, the discovery currents are tested against a final window. This window currently involves a very high current on the second measurement (8.8 volts) and a very low current on the third measurement (4.4 volts). This signature indicates a capacitor on the port that cannot be charged up to 8.8 volts, and that then is not discharged down to 4.4 volts. To characterize this device, if capacitive discovery is enabled, the TPS2384 is put in voltage measurement mode, and the state is set to *pre_capacitance*.

If none of the discoveries take place, the port is flagged as not discovered, the port is reset, and the state is set to *idle*.

1.2.2.4 Pre_capacitance State

The *pre-capacitance* state reads and stores the results of the voltage measurement started by the previous state. If the voltage is low enough, and the port and capacitive discovery are still enabled, it starts a capacitance measurement. It sets the TPS2384 up in a mode which puts out a fixed current and measures voltage. It sets the state to *capacitance*.

If the voltage is not low enough or something is disabled, the port is reset and set back to *idle* mode.

1.2.2.5 Capacitance State

If the port and capacitive detection are still enabled, the *capacitance* state reads in the voltage result from the constant current. This is then subtracted from the pre-capacitance voltage to get a charge rate. If this charge rate is within the window of the PD signatures, the device is considered to be discovered. If powerup is enabled, ramp up is started, and the state is set to *ramp_up_power*. If not, the state is set to *discovered_legacy*.

If something is disabled, or if the voltage difference is outside the window, the device is labeled as not discovered, the port is reset, and the state is set to *idle*.

1.2.3 Discovered, Classify

The discovered states are provided because of the tight control that power management has over PDs in the firmware. In general, the ports are not enabled for powerup until they are discovered. It is necessary for the firmware to set the discovered bit, and for the power manager to recognize this and set the powerup enabled bit. The discovered states permit one port cycle to pass to enable this to happen. All of these states reset the port and return it to the idle state if it is disabled.

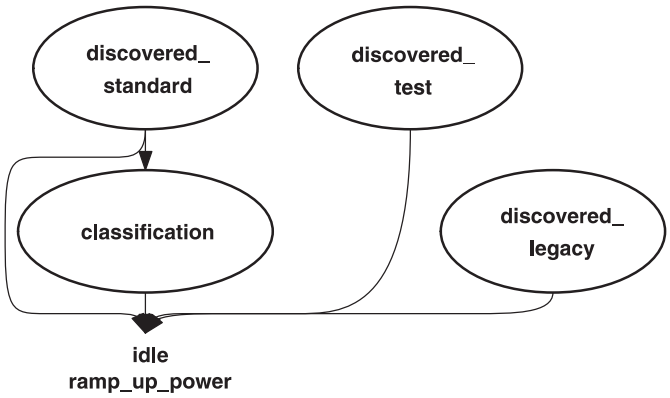

Figure 1-5. Discovered, Classify States Detail

1.2.3.1 Discovered_standard State

The *discovered_standard* state is reached, not surprisingly, from a standard discovery. Like most states, it checks first to see if the port is enabled. If it is, and if powerup and classification are enabled, it starts the classification process on the port, and sets the state to *classification*. If classification is disabled, but powerup is enabled, it starts the ramp up on the port, and sets the state to *ramp_up_power*. Note that these steps are only taken if no other FET on the TPS2384 is operating in the linear region.

Even though there is a classification enable bit, the existing software does not clear it, so the *discovered_standard* state should always advance to the *classification* state.

1.2.3.2 Discovered_test State

The *discovered_test* state is very simple. If the port is enabled for powerup and no other FET on the device is in linear mode, the ramp up process is started, and the state is set to *ramp_up_power*. If not, the port is reset and the state is set to *idle*.

1.2.3.3 Discovered_legacy State

The *discovered_legacy* state is entered either after a legacy or capacitive discovery. It is very similar to the *discovered_test* state. The only difference is that the discovered_test state puts the PORT_STATUS_TEST_MODE value into the temporary register. The *ramp_up_power* state puts this value into the port status memory. For *discovered_legacy* the value has been put in during previous states, and reflects either legacy or capacitive discovery.

1.2.3.4 Classification

The *classification* state is entered after discovery if powerup and *classification* are enabled. *Classification* is deliberately put after powerup enable because it can draw a fair amount of power itself – about 100 milliamps or 4.8 watts worst case. *Classification* reads the classification current from the TPS2384 and determines the PD class. Then it starts power ramp up, sets the state to *ramp_up_power*, and loads the temporary register with PORT_STATUS_POWERED_RESISTIVE_DISCOVERY.

1.2.4 RampUp

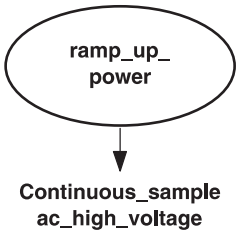

Figure 1-6. Ramp up States Detail

The *ramp_up_power* state is entered from a variety of states. If DC disconnect is enabled, it sets the port up for continuous sample mode and sets the port state to continuous sample.

If AC disconnect is enabled, *ramp_up_power* pauses 1 port cycle if necessary to sync up to the AC disconnect waveform. It then sets the port up for voltage mode and sets the state to *ac_high_voltage*.

In any case, *ramp_up_power* turns off the flag indicating that the FET is in the linear mode, puts the port status from the temporary register into the port status byte, flags that the port is powered up, and puts a nominal 48 volts into the port voltage. This is done because it takes considerable time to measure the voltage in continuous sample mode.

1.2.5 Power Up –DC Disconnect

The power up group of states is discussed in two groups – DC disconnect and AC disconnect.

The TPS2384 has a mode – continuous sample – that provides significant automation for DC disconnect. Whenever the port is powered up, the TPS2384 provides undervoltage, overvoltage and overcurrent protection. When the port is also in continuous sample mode, the TPS2384 provides two other functions as well; an automatic IEEE-compatible DC disconnect function, and automated sequencing between current and voltage reads.

Because of the high level of support provided by the TPS2384, the main loop for DC disconnect is very simple. Each port is processed in a simple sequence starting with the lowest number port and ending with the highest.

This is referred to as a port cycle.

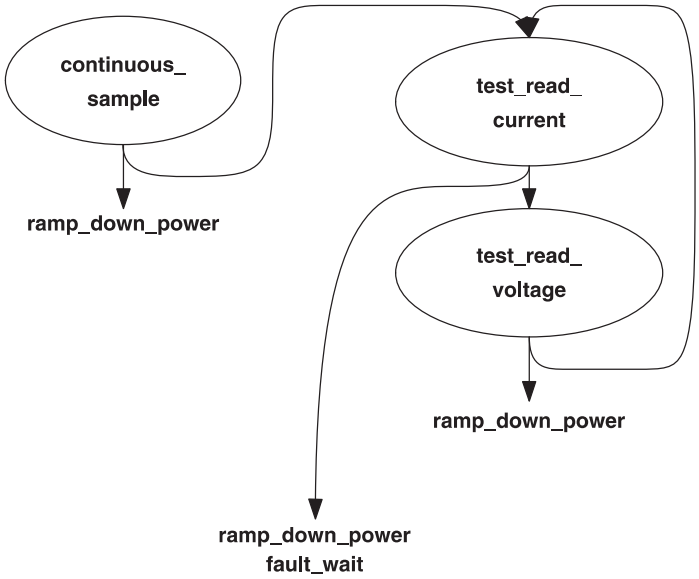

Figure 1-7. DC Disconnect States Detail

1.2.5.1 Continuous Sample

The *continuous_sample* state is the normal state for a powered-up port on the TPS2384. It always checks to verify that the port is still enabled for powerup. If not, it ramps down the port and sets it to *ramp_down_power*.

If the port is in test mode, the *continuous_sample* routine starts a current read and sets the state to *test_read_current*.

Normally, the *continuous_sample* state reads port current. Periodically, but much less frequently, it reads port voltage. Always, it calculates port power. If the power is over the port power limit, it ramps down the port and flags an overlimit fault.

Normally, the *continuous_sample* mode is stable for a long time, as long as the PD is powered up.

The *continuous_sample* mode also reads the port status, and sets the port state to *control_state_fault_wait* in order to further distinguish fault types.

1.2.5.2 Test_read_current

The *Test_read_current* state is provided to override the automatic DC disconnect feature in the TPS2384. In the IEEE mandated test mode, port power is supposed to be maintained regardless of the presence or absence of a PD on the port. The automatic DC disconnect feature in continuous sample mode on the TPS2384 would defeat this. However, it is still necessary, even in test mode, to read current and voltage and perform power management.

The *test_read_current* state first checks to see that the port is still enabled for powerup and is still in test mode. If these criteria are not met, it starts the ramp down process, and sets the port state to *ramp_down_power*.

If the criteria are met, it reads and stores the current value from the TPS2384. Next, it checks for a fault indication from the TPS2384. If it gets one, it sets the state to *fault wait*, and sets the port status to UV/OV fault.

Test_read_current also calculates the power consumption for the port every time through. Since it is a test mode, however, it does not perform an overlimit calculation on the port power limit.

If there is no fault, the firmware starts a voltage measurement process on the TPS2384 and set the port state to *test_read_voltage*.

1.2.5.3 Test_read_voltage

Test_read_voltage is very similar to test_read_current, except that it reads voltage instead. It does not test for a fault. Since the TPS2384 shuts the port down automatically upon fault detection, it is not necessary to check every single port cycle.

If all is well, *test_read_voltage* sets up for a current measurement and sets the state to *test_read_current*.

1.2.6 Power Up – AC Disconnect

While the TPS2384 provides an interface to drive the FETs for a square wave based AC disconnect, it provides no help with timing that square wave or with detecting disconnect. All these functions must be provided by the firmware on the MSP430.

The need to provide these functions, as well as the speed limitations of the I^2C interface to the TPS2384, make the main loop for AC disconnect much more complex than that for DC disconnect.

The software supports up to 6 modules, each with its own AC disconnect waveform. The frequency is 20 Hz., so each half cycle takes 25 milliseconds. The module waveforms are usually 25/n milliseconds apart, when n is the number of installed modules.

For example, in a system with 3X Modules (2X8), the waveforms are spaced 8.3333 milliseconds apart, as shown in Figure 1-8.

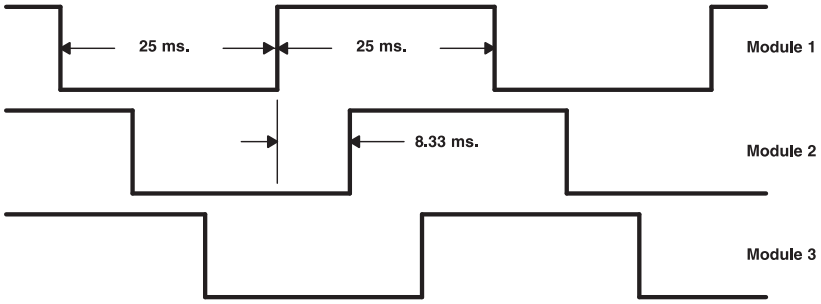

Figure 1-8. AC Disconnect Waveforms

The TPS2384 has an integrating A/D converter that has a maximum conversion time of 21 milliseconds. For this reason, the software is structured so that it starts as many conversions as possible in the first 4 milliseconds of the half cycle on each module. For more information on the AC disconnect timing, consult the source listings.

Figure 1-9 shows the AC disconnect states.

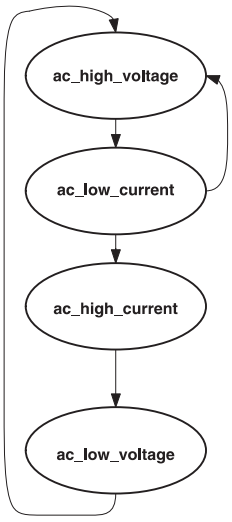

Figure 1-9. AC Disconnect States Detail

1.2.6.1 AC_high_voltage

AC_high_voltage first checks to see if the port is still enabled for powerup. If not, it starts a ramp-down and goes to the *ramp_down_power* state. Next it adds the port power to the system power accumulators.

Next it checks the *ac_high* variable to make sure that the AC disconnect signal is high. If the signal is not high, the state just waits another cycle for it to be high so that it can do a correct measurement.

Next, it reads the voltage value from the TPS2384 and stores it in the temporary register for the port. Then it starts another voltage-read cycle for the port and sets the state to ac_low_current.

The names of states in AC disconnect describe the operation for which they prepare parts of the system, not the operation that they complete. So the *ac_high_voltage* state starts the measurement of voltage with the AC signal high. The voltage it reads in is from the AC low phase.

1.2.6.2 AC_low_current

The *ac_low_current* state performs the usual checks for powerup enable and AC phase. These are also performed for the rest of the AC states.

Next, it reads in the voltage from the TPS2384. This voltage is the high AC phase voltage. It averages the high and low voltages and saves the average as the port voltage. It then calculates the port power. If the port power is over the limit, it starts ramp-down, and sets the state to *fault_ramp_down*.

If the power is acceptable, it then calculates the difference between the high and low phase voltages. If the difference is above a threshold, the PD has probably been removed. If a difference above a threshold is found, the state is set back to *ac_high_voltage* and not to *ac_high_current*. So the AC-disconnect sequence is shortened into only two states until the next AC-disconnect sequence entry.

The software waits for a specific number (6 times) of high voltages in succession before ramping down the port and going to *fault_ramp_down* state. This also ensures that *Green-PDs* with modulated current consumption are not disconnected.

Ac_low_current starts a current measurement and sets the state to *ac_high_current*.

1.2.6.3 AC_high_current

Ac_high_current is very similar to *ac_low_current*. First, it performs the usual checks for powerup-enable and AC phase. Then it reads the content of the port status register. If any fault is detected when reading the status, the state *control_state_fault_ramp_down* is set. Otherwise, it initiates the high current measurement and sets the state to *ac_low_voltage*.

1.2.6.4 AC_low_voltage

AC_low_voltage reads the high current measurement. This value is used for the port power and port power limit check. If that within limits, it starts a voltage measurement and sets the state to *ac_high_voltage*. This completes the AC-disconnect cycle.

1.2.7 Ramp Down/Fault States

The Ramp-Down/Fault states are entered from the power-on states. The only way for a ramp down to occur without a fault is either for the port to be disabled, or for power management to power down the port. An AC or DC disconnect is considered to be a fault.

The Ramp Down/Fault states are shown in Figure 1-10.

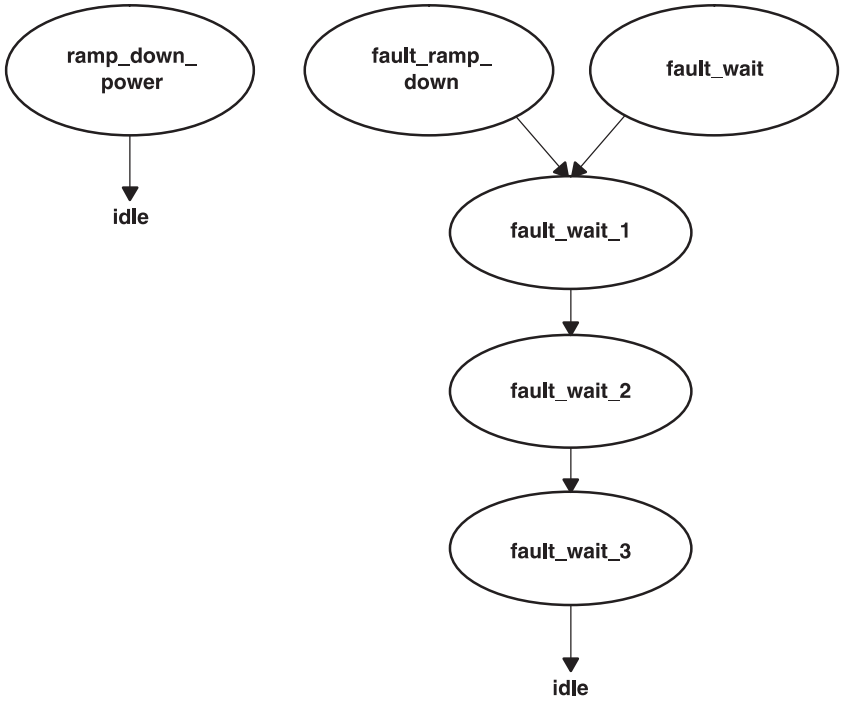

Figure 1-10. Ramp Down/Fault States Detail

1.2.7.1 Ramp_down_power

The *ramp_down_power* state is very simple. It occurs when either the port currently on is disabled or port powerup is disabled. It simply resets the port, clears various flags and values such as power and voltage, and sets the state to *idle*. Since it is shutting down the port, there is no need to check for enable or any other inputs.

1.2.7.2 Fault_ramp_down

Fault_ramp_down occurs upon any fault from the TPS2384, or when there is an AC disconnect event, or when a PD is consuming more power than a port can support. It also has no checks for control inputs. It follows states that do not have time to start a ramp-down function. Some states spend too much time doing reads and writes, and would overflow their time slot if they tried to start a ramp down as well. Like *ramp_down_power*, it clears power, voltage, and current, and various flags. It advances to *fault_wait_1*.

1.2.7.3 Fault_wait

Fault_wait occurs when there is any fault (including a disconnect event) from a TPS2384. This applies to both DC disconnect mode and test mode. It is designed to follow states that do not have time left to reset the port. Several states take so much time reading from and writing to the TPS2384 that they would run out of time in their time slot.

Fault_wait resets the port, clears the powerup parameters and flags, and sets the state to *fault_wait_1*.

1.2.7.4 Fault_wait_1

Fault_wait_1 updates fault status, and sets the value in *general_count* as a delay count according to the specific type of fault encountered. (Delay countdown is handled by power management.) Then it sets the next state to *fault_wait_2*.

1.2.7.5 Fault_wait_2

Fault_wait_2 starts a temperature measurement, then sets the next state to *fault_wait_3*.

1.2.7.6 Fault_wait_3

Fault_wait_3 reads a temperature-measurement result, and counts down *general_count* by one. If the delay time has not expired, it continues to start another temperature measurement without changing the state; otherwise, it sets the next state to *idle*, and resets the port in the TPS2384.

1.3 Host Interface Protocol

The PoE system solution includes the MSP430 microcontroller on which the firmware runs, as well as up to 12 TPS2384 power controllers. This system interoperates with a host over an RS-232 interface. In the future, this interface can also be an I^2C bus.

In order to interoperate with the host, a well-defined protocol exists between the MSP430 microcontroller and the host. This is referred to as the *Host Interface Protocol*.

The Host Interface Protocol consists of a set of commands that allow the host processor system and PoE system to interact. Each of these commands defines a data format that allows this exchange of information between the systems.

1.3.1 Command Overview

The commands comprising the Host Interface Protocol are shown in Table 1-2.

Table 1-2. Host Interface Protocol Commands

Command	Code	Description	Direction
Reset	0x52	Resets the MSP430 firmware and system hardware into a known state. The MSP430 responds with an Acknowledge message.	Host to PoE
System Write	0x05	Writes parameters that control system-level functionality. The MSP430 responds with an Acknowledge message.	Host to PoE
Save/Restore Configuration	0x06	Saves parameters to the flash memory, or restores default parameters. The MSP430 responds with an Acknowledge message.	Host to PoE
Program Controller	0x07	Initiates software download of a new image to the MSP430. The MSP430 responds with an Acknowledge message.	Host to PoE
Port Write	0x80 – 0xAF (1)	Writes parameters that control port-level functionality. The MSP430 responds with an Acknowledge message.	Host to PoE
Information Request	0xBA	Requests information from MSP430 such as system status, port status, etc. The MSP430 responds with a System Read, Power Read, Port Read, All Ports Status or All Ports Enabled Message.	Host to PoE
Clear Application	0xF0	Erase the PoE application program. The MSP430 responds with an Acknowledge message.	Host to PoE
Clear Information	0xF1	Erase the PoE configuration information. The MSP430 responds with an Acknowledge message.	Host to PoE
Receive Data Block	0xF2	Send a data block for programming into the Flash memory. The MSP430 responds with an Acknowledge message.	Host to PoE
Acknowledge	0xBA	Acknowledges receipt of a Reset, System Write, Save/Restore Configuration, Program Controller or Port Write message from the Host, or an Acknowledge of an error detected in any requesting message.	PoE to Host
System Read	0x05	System-level parameters sent in response to an Information Request message.	PoE to Host
Power Read	0x08	System power information sent in response to an Information Request message.	PoE to Host
System Info	0x09	Basic boot-level parameters sent in response to an Information Request message asking for system info parameters.	PoE to Host
Port Status	0x10 – 0x13	Status of a group of ports, sent in response to an Information Request message asking for all ports status.	PoE to Host
Port Enables	0x20	Enable status of all ports, sent in response to an Information Request message.	PoE to Host
Port Read	0x80 – 0xAF	Port level parameters, sent in response to an Information Request message.	PoE to Host

(1) The Port Write code contains both the code and the port which should be written.

The interface for the Host Interface Protocol is defined in the protocol.h header file.

1.3.1.1 Protocol Format

The protocol consists of a message code (as described in Table 1-2), 0 or more data bytes specific to the message code, and a 2-byte checksum field.

The checksum is the 16-bit sum of all the previous bytes of the message. This is generated by the transmitter and validated by the receiver.

The 16-bit (two-byte) data types are stored as follows: the high byte of the data is stored in the first storage location (closer to the message code) and the low byte of the data is stored in the second storage location.

1.3.1.2 Protocol Validation

The MSP430 controller monitors the incoming message data to ensure integrity within all communications messages. The PoE controller verifies the following:

- The checksum is correct
- The length of the message is correct for the associated command
- Invalid data is not received with the specific command
- The message is received in a timely fashion

Any of these detected errors is recorded and transmitted back to the host controller via an Acknowledge message and the appropriate Response Code.

1.3.2 Transactions: Host Controller to Power Subsystem

The following messages are used for communication initiated by the host controller to the power system (MSP430 and TPS2384).

1.3.2.1 Reset

This message is sent from the host processor to the MSP430 to reset the PoE controller(s). The values of the first 5 bytes spell out the word *RESET*, to ensure that the MSP430 does not trigger a system reset incorrectly. The MSP430 responds immediately to this message with an *Acknowledge* message. With a full configuration, it can take up to 2 seconds to perform the reset – when it is completed, a System Info message is sent.

Table 1-3. Reset Message Format

Byte	Fleld	Values
1	Reset code	0x52 ('R')
2	Data	0x45 ('E')
3	Data	0x53 ('S')
4	Data	0x45 ('E')
5	Data	0x54 ('T')
6-7	Checksum	0x0183

1.3.2.2 System Write

This message is sent from the host processor to the MSP430 to write system level parameters in the MSP430 that control the TPS2384 power controller(s). The MSP430 responds to this message with an *Acknowledge* message.

If the Modify bit is set for a particular parameter, the parameter data is applied to the system parameter. If the Modify bit is not set for a particular parameter, the parameter data is ignored.

The *Knockoff* bit, when set, allows Power Management to allow a higher priority device, when plugged in, to take precedence over and power down a lower priority device.

If the Start bit is set, it starts processing, and locks certain bits as described below.

The Modify Enable allows the Number of Module, Number of Ports and the 1st Port Number on Modules to be changed. The number of modules refers to the number of AC disconnects circuits in the system. For example a 24 port system might have 2 AC disconnect circuits, each servicing 12 ports. So the Number of Modules for that system would be 2. The Number of TPS2384 in the system refers to the number of TPS2384's in the system. There are 4 ports per chip so in the 24 port system there there would be 8 TPS2384's in the system.

The *1st Port Number on 1st Module* is not available, as this is a fixed address (of 0) and can not be changed. The *Number of Modules* must reflect that there is always at least 1 module. The next unused *1st Port Number of X Module* must reflect the next unused address, which indicates what the last address of the previous module is. For example, on a 20-port, 2-module system, you might have *1st Port Number on 1st Module = 12* and *1st Port Number on 2nd Module* = 20. The port numbers are zero-based, so port 20 indicates physical port 21.

The changes to the Port and Module parameters do not take affect until a Save/Restore Configuration message is received, with at least the *System Settings* requested.

Table 1-4. System Write Message Format

Byte	Field	Values
1	System Write code	0x05
2	Knockoff , AC Disconnect, Detection Type and Start parameters: • Bit 0: Modify *Knockoff* • Bit 1: Modify *AC Disconnect* • Bit 2: Modify *Detection Type* • Bit 3: Modify *Start Control* Knockoff: • Bit 4: Knockoff (0-enabled, 1- disabled) AC Disconnect: • Bit 5: AC Disconnect (0-DC, 1-AC) Detection Type: • Bit 6: Detection Type (0-resistive, 1-capacitive) Start Control: • Bit 7: Start (0-OFF, 1-ON)	Varies
3-4	Maximum Power available when only one supply is working (watts)	Varies – 0xFFFF represents no change
5-6	Maximum Power available when both supplies are active (watts)	Varies – 0xFFFF represents no change
7	User-defined label for the PoE controller	Varies
8	Port and Module parameters: Modify Enables: • Bit 0: Modify *Number of Modules*, *Number of Ports*, and *1st Port Number on Module X* Number of Moduels: • Bits 1-3: Number of Modules (1 to 6) Number of Ports: • Bits 4-7: Number of Ports (in multiples of 4: 1-4 ports, 2-8 ports, ... 12-48 ports)	Varies
9	1st Port Number on 2nd Module	Varies
10	1st Port Number on 3rd Module	Varies
11	1st Port Number on 4th Module	Varies
12	1st Port Number on 5th Module	Varies
13	1st Port Number on 6th Module	Varies
14	Unused	Varies
15-16	Checksum	Varie

1.3.2.2.1 Start up Sequence

After a hardware or software reset command, the firmware:

- Initializes the MSP430
- Asserts reset on the TPS2384s, disabling all activity on the PoE ports
- Sends the System Read message to the host (only valid on RS-232 interface – the I^2C version does not automatically send this message, it must be polled for by the host)
- Handles all host interface protocol messages

After initializing the port and system configurations, the host sends a System Write command with the start bit set. This causes the firmware to:

- Lock in AC disconnect (and start bit)
- Initialize the TPS2384s (≈ 5 seconds)
- Start running the PoE ports
- Start again to handle all the interface protocol messages

The only way to change the AC disconnect is to reset the MSP430, either with a hardware reset, or with a Reset command.

1.3.2.3 Port Write

This message is sent from the host processor to the MSP430 to write port-level parameters in the MSP430 that control the TPS2384 power controllers. The MSP430 responds to this message with an *Acknowledge* message.

The ability to set the Logical Port Number is not available when sending this message addressed to *All Ports*. Setting of the Logical Port Number is mutually exclusive of setting any other parameter using the Port Write message. When setting the Logical Port Number, the Port Write code always refers to the physical port number, regardless of any other previous Logical Port setting for the port number. The Logical Port Number does not take affect until a Save/Restore message is received with the Logical Port Numbering parameter set for saving. The Physical and Logical Port Numbers must be within the configured *Number of Ports* that is part of the System Write message.

When not setting the Logical Port Number, the Port Write code refers to the remapped (if previously set) logical port, or the physical port (if not previously set).

Table 1-5. Port Write Message Format

Byte	Field	Values
1	Port Write code	0x80 – 0xAF is an individual port (0x80 + port number). 0xB0 represents all ports.
2	Modify Enables: • Bit 0: Modify 'Port Enable' • Bit 1: Modify 'Port Priority' • Bit 2: Modify 'Legacy Support' • Bit 3: Modify 'Power Management' • Bit 4: Modify 'Test Mode' • Bit 5: Modify 'I2C Parameters' • Bit 6: Modify 'Clear Events' • Bit 7: Modify 'Logical Port Number'	Varies
3	Port Enable: • Bit 0: Port Enable (0- disabled, 1 enabled, disabled is default) Port Priority: • Bits 1-2: Port Priority (1-critical, 2-high, 3- low, 0 indicates no change) Legacy Support: • Bit 3: Legacy Support (0-disabled, 1-enabled, disabled is default), • Bit 4: Capacitive Support (0-disabled, 1-enabled, disabled is default) Power Management: • Bit 5: Power Limit from Classification (0-disabled, 1- enabled, disabled is default) • Bit 6: Power Limit for Management (0- disabled, 1-enabled, disabled is default) Test Mode: • Bit 7: Test Mode (0- Auto, 1-Test, Auto is default),	Varies
4	I^2C Parameters: • Bits 0-4: Address (1 to 31) • Bit 5-6: Bus (1 to 3, default is 1) Clear Events: • Bit 7: Clear Events (1-clears Overload and Underload flags)	Varies
5	Logical Port Number (used for all future protocol messages when per-port information is requested)	Varies
6-7	Maximum Power (milliwatts)	Varies – 0xFFFF represents no change, (default power is 15.4W)
8-9	Checksum	Varies

1.3.2.4 Save/Restore Configuration

This message is sent from the host processor to the MSP430 to record system and channel-level parameters to flash memory, and uses that information through subsequent resets of the device. This message is also sent from the host processor to the MSP430 to restore the initial default values for all configuration information. The MSP430 responds to this message with an Acknowledge message.

If the Modify bit is set for a particular parameter, the parameter data will be applied to the system parameter. If the Modify bit is not set for a particular parameter, the parameter data will be ignored.

The System Settings saved are:

- Number of ports, modules, first port number in each module
- Port enabled state, power priority, power limits

  It may take up to 2 seconds to record this information; therefore the host processor needs to wait before resuming transactions with the MSP430. The PoE does not reset after this operation.

The Logical Port Numbering settings saved are:

- Logical port numbers sent in Port Write message

  All the logical port numbers assigned up to this point are recorded in the flash memory, and the PoE then resets.

The restoring of all initial default values occurs after the Acknowledge message is sent and then the MSP430 writes them to flash memory and resets.

Table 1-6. Save/Restore Configuration Message Format

Byte	Field	Values
1	Save/Restore Configuration code	0x06
2	Modify Enables: • Bit 0: Modify 'Save Configuration' • Bit 1: Reserved • Bit 2: Reserved • Bit 3: Modify 'Restore Configuratjion' Save Configuration: • Bit 4: System Settings (1-save) • Bit 5: Logical Port Numbering (1- save) • Bit 6: Reserved Restore Configuration: • Bit 7: Restore Default Data (1- restore)	Varies
3-4	ChecksumChapter 7	Varies

1.3.2.5 Program Controller

This message is sent from the host processor to the MSP430 to initiate a software download of a new image to the MSP430. See the section on *Software Download* for more information on this process.

Table 1-7. Program Controller Message Format

Byte	Field	Values
1	Program Controller code	0x07
2	Check code	0xAA
3	Check code	0x55
4-5	Checksum	0x-106

1.3.3 Information Request

This message is sent from the host to the MSP430 to request information from the MSP430 such as system status, port status, etc. The MSP430 responds with an informational response. No separate Acknowledge message is sent by the MSP430.

Table 1-8. Information Request Message Format

Byte	Field	Values
1	Information Request code	0xBA
2	Message Code	Message Code: 0x05: System Read 0x08: Power Read 0x09: System Info 0x10: Port Status (ports 0–11) 0x11: Port Status (ports 12–23) 0x12: Port Status (ports 24–35) 0x13: Port Status (ports 36–47) 0x20: Port Enable Status (all ports) 0x80: 0xAF – Port Read
3-4	Checksum	Varies

1.3.4 Transactions: Power Subsystem to Host Controller

The following responses are sent by the MSP430 in request to a message from the host.

1.3.4.1 Acknowledge

This is a message from the MSP430 to the host that acknowledges receipt of a Reset, System Write, Port Write, Save/Restore Configuration, Program Controller, Clear Application, Clear Information, or Receive Data Block message from the Host, or an acknowledge of an error detected in the requesting message.

Table 1-9. Acknowledge Message Format

Byte	Field	Values
1	Acknowledge Code	0xBA
2	Response Code	Response Code: 0x00: Success 0x01: Bad Checksum 0x02: Message Too Long 0x03: Message Not Recognized 0x04: Invalid Data In Message 0x05: Message Timed Out 0x06: Programming Error
3-4	Checksum	Varies

Bad Checksum:	The calculated checksum of the received message does not match that of the received checksum.
Message Too Long:	The received message count is longer than the expected count for the specific command (based on the Command Code).
Message Not Recognized:	The Command Code received is not part of the accepted list of Command Codes.
Invalid Data In Message:	Data associated with the specific command does not meet the specification of the command.
Message Timed Out:	More than 100 milliseconds has elapsed between receiving one byte of the message and the next.
Programming Error:	An error occurred while attempting to write to the FLASH during the programming procedure, or the correct sequence was not followed.

1.3.4.2 System Read

This is a message containing system-level parameters sent from MSP430 to host. This message is sent in response to an Information Request message asking for system level parameters.

Table 1-10. System Read Message Format

Byte	Field	Values
1	System Read Code	0x05
2	Reserved • Bit 0 Factory Defaults: Boot Message: Knockoff: AC Disconnect: Detection Type: Start Control:	Reserved • Bit 0: Reserved Factory Defaults: • Bit 1: Kernel Status (1-using factory default values) Boot Message: • Bit 2: Boot Time Message(0 – normal runtime, 1-boot time) Knockoff: • Bit 4: Knockoff (0-enabled, 1- disabled) AC Disconnect: • Bit 5: AC Disconnect (0-DC, 1- AC) Detection Type: • Bit 6: Detection Type (0-resistive, 1-capacitive) Start Control: • Bit 7: Start (0-OFF, 1-ON)
3-5	Serial Number	Varies
6	Chip Revision	Varies
7	Chip ID	Chip ID: 0x00: TPS2383 0x01: TPS2383A 0x02: TPS2384 0x03-0xFF: reserved
8	Firmware Version:	Firmware Version: • Bits 0-2: Patch • Bits 3-5: Minor • Bits 6-7: Major (0-1, 1-2, 2-3, 3-4)
9	Port and Module parameters: Reserved • Bit 0 Number of Modules: Number of Ports:	Port and Module parameters: Reserved • Bit 0: reserved Number of Modules: • Bits 1-3: Number of Modules (1 to 6) Number of Ports: • Bits 4-7: Number of Ports (in multiples of 4: 1-4 ports, 2-8 ports, ... 12-48 ports)
10	1st Port Number on 2nd Module	Varies
11	1st Port Number on 3rd Module	Varies
12	1st Port Number on 4th Module	Varies
13	1st Port Number on 5th Module	Varies
14	1st Port Number on 6th Module	Varies
15	User-defined label for the PoE controller	Varies
16-17	Checksum	Varies

1.3.4.3 Power Read

This is a message containing system-power information sent from MSP430 to host. This message is sent in response to an Information Request message asking for system power.

Table 1-11. Power Read Message Format

Byte	Field	Values
1	Power Read code	0x08
2-3	Actual power consumption (watts)	Varies
4-5	Maximum shutdown voltage (decivolts)	Varies
6-7	Minimum shutdown voltage (decivolts)	Varies
8-9	Main supply voltage (decivolts)	Varies
10	Power Sourcevoltage (decivolts)	Power Source: 0x00 – power source 2 (both supplies) 0x01 – power source 1 0x02 – other power source 0x03 – no power source
11-12	Maximum power available (watts)	Varies
13-14	Checksumpower available (watts)	Varies

1.3.4.4 System Info

This is a message containing system-info parameters sent from MSP430 to host. This message is sent in response to an Information Request message asking for system info parameters.

Table 1-12. System Info Message Format

Byte	Field	Values
1	System Read code	0x09
2	Boot/Kemel Status: • Bits 0-1 Reserved: • Bits 2-7	Boot/Kemel Status: • Bit 0: Boot Error (1-needs download of new image) • Bit 1: Code Type (0-PoE running, 1-BSL running) Reserved: • Bits 2-7
3	Firmware Version:	Firmware Version: • Bits 0-2: Patch2 • Bits 3-5: Minor5 • Bits 6-7: Major (0-1, 1-2, 2-3, 3-4)
4-5	Checksum	Varies

1.3.4.5 Port Read

This message contains port-level parameters sent from the MSP430 to the host. This message is sent in response to an Information Request message asking for port level parameters.

Table 1-13. Port Read Message Format

Byte	Field	Values
1	Port Read code	0x80 – 0xAF (0x80 + port number)
2	Control Parameters	Port Enable: • Bit 0: Port Enable (0-disabled, 1 enabled) Port Priority: • Bits 1-2: Port Priority (1-critical, 2-high, 3-low, 0 indicates no change) Legacy Support: • Bit 3: Legacy Support (0- disabled, 1-enabled) • Bit 4: Capacitive Support (0- disabled, 1-enabled) Power Management: • Bit 5: Power Limit from Classification (0-disabled, 1- enabled) • Bit 6: Power Limit for Management (0-disabled, 1- enabled) Test Mode: • Bit 7: Test Mode (0-Auto, 1-Test)
3	I^2C Parameters:	I^2C Parameters: • Bits 0-4: Address (1 to 31) • Bits 5-6: Bus (1 to 3, default is 1)
4-5	Maximum Power (milliwatts)	Varies (is 'Maximum Power' from Port Write message)
6	Status	Status: 0x00: Disabled 0x01: Detection active 0x02: Powered through IEEE resistive detection algorithm 0x03: Powered through capacitive detection algorithm 0x04: Overload fault 0x05: Underload fault 0x07: Power Managed 0x09: Invalid PD 0x0A: Limit Overload fault 0x0C: Unable to reset TPS2384 device 0x0D: Unable to initialize TPS2384 device 0x0E: Power with Cisco Legacy PD 0x0F: Un-initialized 0x10: Status is nonexistent 0x12: Thermal fault 0xFF: Undefined status
7	PD Classification: • Bits 0-2 Reserved: • Bits 3-5 Events: • Bits 6-7	PD Classification: • Bits 0-2 0: Class 0 (15.4W) 1: Class 1 (4W) 2: Class 2 (7W) 3: Class 3 (15.4W) 4: Class 4 (15.4W) Reserved: • Bits 3-5 Events: • Bit 6: Underload occurred • Bit 7: Overload occurred
8-9	Actual Voltage (decivolts)	Varies
10-11	Actual Power (mW)	Varies
12-13	Actual Current (mA)	Varies
14-15	Checksum	Varies

1.3.4.6 Ports Status

This message contains status of a group of ports sent from the MSP430 to the host. This message is sent in response to an Information Request message asking for port status.

The Port Status value is the same as defined in the Port Read ('Status' byte) message.

Table 1-14. All Ports Status Message Format

Byte	Field	Values
1	Port Status code	Status: 0x10 – Status of ports 0-11 0x11 – Status of ports 12-23 0x12 – Status of ports 24-35 0x13 – Status of ports 36-47
2	Status	Status of port 0, 12, 24 or 36
3		Status of port 1, 13, 25 or 37
4		Status of port 2, 14, 26 or 38
5		Status of port 3, 15, 27 or 39
6		Status of port 4, 16, 28 or 40
7		Status of port 5, 17, 29 or 41
8		Status of port 6, 18, 30 or 42
9		Status of port 7, 19, 31 or 43
10		Status of port 8, 20, 32 or 44
11		Status of port 9, 21, 33 or 45
12		Status of port 10, 22, 34 or 46
13		Status of port 11, 23, 35 or 47
14-15	Checksum	Varies

1.3.4.7 Port Enables

This is a message containing the enabled status of all ports from the MSP430 to the host. This message is sent in response to an Information Request message asking for Port Enable status.

Table 1-15. Port Enables Message Format

Byte	Field	Values
1	Port Enables code	0x20
2	Port Mask (ports 0–7)	• Bit 0 – Port 0 enable • Bit 1 – Port 1 enable • Bit 2 – Port 2 enable • .... • Bit 7 – Port 7 enable
3	Port Mask (ports 8–15)	• Bit 0 – Port 8 enable • Bit 1 – Port 9 enable • Bit 2 – Port 10 enable • .... • Bit 7 – Port 15 enable
4	Port Mask (ports 16–23)	• Bit 0 – Port 16 enable • Bit 1 – Port 17 enable • Bit 2 – Port 18 enable • .... • Bit 7 – Port 23 enable
5	Port Mask (ports 24–31)	• Bit 0 – Port 24 enable • Bit 1 – Port 25 enable • Bit 2 – Port 26 enable • .... • Bit 7 – Port 31 enable
6	Port Mask (ports 32–39)	• Bit 0 – Port 32 enable • Bit 1 – Port 33 enable • Bit 2 – Port 34 enable • .... • Bit 7 – Port 39 enable
7	Port Mask (ports 40–47)	• Bit 0 – Port 40 enable • Bit 1 – Port 41 enable • Bit 2 – Port 42 enable • .... • Bit 7 – Port 47 enable
8-9	Checksum	Varies

Initialization

2.1 Overview

The PoE program on the MSP430 supports a very flexible PoE system by allowing many different configuration parameters for the system. Most of these parameters are dynamic, and can be changed at any time. Some of these must be set before the application on the MSP430 is started.

2.2 Sequence of Events

In general, the process for system initialization is:

- Release the MSP430 from reset
  - Boot loader application starts, checks for valid PoE image in flash
  - PoE image begins and initializes microcontroller and restores default or flash values for necessary parameters
  - TPS2384 devices are held in reset at this point
- Host program sends information to the MSP430 at this point:
  - System Write:
    - Disconnect policy (AC or DC)
    - Knockoff (enabled/disabled)
    - Detection type (resistive or capacitive)
    - Power Supply 1 and 2 values (in watts)
    - Number of ports, modules, first port number in each module
    - User-defined private label
  - Port Write (one command for each of the available ports):
    - State (enabled/disabled)
    - Priority (low, high, critical)
    - Legacy support (legacy, capacitive)
    - Power Management (power limit from classification, power limit for management)
    - Test mode (auto/test)
    - I^2C configuration
    - Maximum power
  - Port Write (one command for each of the available ports):
    - Logical port remapping
  - Save Configuration
    - System Settings
    - Logical Settings (if Port Remapping occurred)
- Host program then 'starts' the application:
  - System Write
    - Start
  - MSP430 releases all TPS2384 devices from reset,
  - Initializes them, and can begin the application in earnest, powering ports, etc.
  - At this point, most commands can be given. (Normal operation)

2.3 Detail

The PoE system must have a basic set of information configured – or must use the defaults – before the start signal is given to the application. The defaults are constructed to allow the TPS2384 reference design to work without any configuration changes.

The information that is needed by the application before it begins is:

- AC Disconnect or DC Disconnect (AC default)
- Number of ports in the system (12 default)
- Number of modules in the system (1 default)
- I^2C address for each channel/TPS2384 in system
- Starting port number for each module in system (port 0 for module 0 default, no others set)
- Power supply 1 and Power supply 2 values.

AC and DC Disconnect

Disconnect is a method to detect a disconnection of a PD from a PoE-enabled port on a PSE. This functionality is defined by the IEEE 802.3af standard. This method is required to determine if and when a load from a PD is no longer present, and to disable power within a short amount of time.

The purpose of Disconnect is to remove power from a port before a new device can be inserted into the powered port. This is essential to prevent the new device from being damaged if it cannot accept power from the PSE.

A valid PD connected to a PSE is detected and powered with a nominal 48 V. When this PD is disconnected from that port, it is essential to prevent a new device from receiving power before a valid detection and authentication process is completed. According to the IEEE 802.3af standard, the process of removing power must be completed within 300-400 ms, when a valid PD is disconnected. This value takes into consideration the fastest predicted time that it would take a person to unplug a PD from a port and plug a new device into that same port.

The IEEE 802.3af standard allows this power removal to be implemented by one or both of the following methods:

DC Disconnect

AC Disconnect

The disconnect method is a system-level parameter for the firmware for the TPS2384 – the same disconnect method is used for all ports on all TPS devices in the system. The disconnect method can be changed via either the *GUI* or through the *Host Interface Protocol*, at the system level. The default operation of the firmware uses AC Disconnect.

3.1 DC Disconnect

DC Disconnect monitors DC current through the port terminals and removes power when the current drops below a certain value for a certain period of time (see the IEEE 802.3af standard for detail on this function). The TPS2384 PSE provides DC Disconnect support in the PSE, without software intervention.

3.2 AC Disconnect

AC Disconnect monitors the AC impedance on the port terminals and removes power when the impedance rises above a certain value, for a certain period (for details, see the IEEE 802.3af specification).

Unlike DC Disconnect, AC Disconnect is supported in firmware on the MSP430 by monitoring impedance on ports.

AC Disconnect is the default disconnect policy of the firmware.

Power Management

4.1 Overview

PoE Power Management methods are used to manage the distribution and prioritization of PDs. Power Management itself is not defined by the IEEE specification. Instead, it is a policy that takes advantage of the PoE specification as it defines such terms as port and system power. Texas Instruments' PoE controllers are supported by firmware that runs on an accompanying microcontroller that implements the Power Management algorithms.

The goals of Power Management in a PoE enabled system are two-fold:

- Power as many PDs as possible
- Limit power cycling of PDs

In many systems, the maximum system power available limits the total number of ports that may be powered. For example, each PD can draw a maximum of 15.4 W, and a 48-port system can draw more than 739 W total system power. If the maximum system power available is less than 739 W, then Power Management becomes necessary so that the available system power may be used in the most efficient manner while meeting the design goals.

4.1.1 Available Power - Pmsys

PS1 and PS2 determine the total available power (Pmsys) with respect to the AC_GOOD and DC_GOOD signals status. Note that PS1 and PS2 are not physical power sources.

The AC_GOOD and DC_GOOD signals are active HIGH, and indicate the availability of power from the correlated PS_AC and PS_DC power supplies which are the physical power sources connected to the PoE system. PS-DC is also called as RPS (Redundant Power Supply).

The values PS1 and PS2 should be set by the PoE-HOST according to the following terms:

PS1 = PS_AC = available power when PS-AC power is on (Only AC_GOOD signal is asserted)

PS2 = PS_AC + PS_DC = Total available power from both PS_DC and PS_AC supplies.

The following table shows the total available power (P_{msys}) with respect to the power-good signals status.

AC_GOOD MSP430 P2.1	DC_GOOD MSP430 P2.0	P_{msys} Total Available Power
0	0	PS_DC (computed as PS2–PS1)
0	1	
1	0	PS1 (PS_AC) Set via the HOST protocol (Supply 1)
1	1	PS2 (PS_AC + PS_DC) Set via the HOST protocol (Supply 2)

4.1.2 Power Management Configuration

The Power Management component allows users to choose the source for the port power limits by configuring two variables. Table 4-1 shows these variables and the associated behaviors.

Table 4-1. Power Management Port Configuration.

Power Limit From Classification	Use Port Power Limit for Power Management	Port Power Limit	Port Power Allocation
No	No	User specified (default 15.4 W)	Measured Power (V × I)
No	Yes	User specified (default 15.4 W)	Port Power Limit
Yes	No	From Classification	Measured Power (V × I)
Yes	Yes	From Classification	Port Power Limit

If a port (or PD) consumes power more than the port power limit, the port will be powered down. The total Power management power is the sum of Port Power Allocation for all the ports on the system.

The maximum power drawn by any PD is always 15.4 W. If any PD draws more than 15.4 W, it is an overload condition and the PD is disconnected.

4.1.3 Disabling Power Management

The Power Management component is integrated into many of the facets of the detection, classification and powering of ports.

To 'disable' power management, users can set the power limits for each supply and for the system sufficiently high so that the actual power draw is never greater than the limit. This will – in effect – prevent the power-management function from being invoked.

4.2 Definitions and Formulas

Table 4-2 defines terms and formulas used in the Power Management algorithm.

Table 4-2. Power Management Terms and Definitions

Term	Definition
P_{csys}	The current total power consumed by PDs, as determined by the power management system. Power consumed will be V × I, but the value of P_{csys} has added to it the allocated power per port. Therefore, P_{csys} = V × I + allocated port power. This value will be used in the power management algorithm for its calculations.
P_{msys}	The maximum total power available for PDs. This power may be provided by one of either 2 system supplies, or by the combination of both system supplies.
P_{d1}	The power value equal to the maximum power one device might draw, 15.4 W, rounded up to 16 watts
$P_{msys-d1}$	$P_{msys} - P_{d1}$ (The maximum total power available for PDs minus the power value equal to the maximum one PD might draw).
$P_{msys-d2}$	$P_{msys} - 2 \times P_{d1}$ (The maximum total power available for PDs minus the power value equal to the maximum two PDs might draw).

4.3 System Parameters

The Power Management algorithm has a number of parameters that it employs in its algorithm. Some of these parameters are configurable and some are constants in the software.

Port priorities are configurable. These parameters can be modified by issuing a command through the *Host Interface Protocol.*

The Power Management software system supports two power supplies. The values of maximum-allowable power draw from each of these supplies are configurable. The availability of the supplies is detected by logic-level signals. The configuration of the total power from these supplies can be directed over the communication interface.

The limits for the various thresholds (P_{msys}, $P_{msys-d1}$, and $P_{msys-d2}$) are not runtime configurable.

These parameters are set during Power Management program compile operation.

4.4 State Definitions

The Power Management algorithm operates as a state machine, whereby the algorithm is a certain state at any given point in time. Table 4-3 shows the state definitions for the algorithm.

Table 4-3. Power Management State Definition Table

State	Definition
NO_CONTROL_NEEDED	Default operational state.
OVERLIMIT	Power demand has exceeded the maximum. Devices have been powered down.
KNOCKOFF	Power Management state where one or more PDs have been powered down so that a higher priority PD may be powered up and yet not exceed the $P_{msys-d1}$.
POWER_UP	A higher priority port has been powered up.
LIMITING	Power is being limited – entered from various states.
REENABLING	Power demand went below $P_{msys-d2}$, and a port was powered up.

4.5 Design Flow

The Power Management algorithm is shown below, in the form of a flow chart.

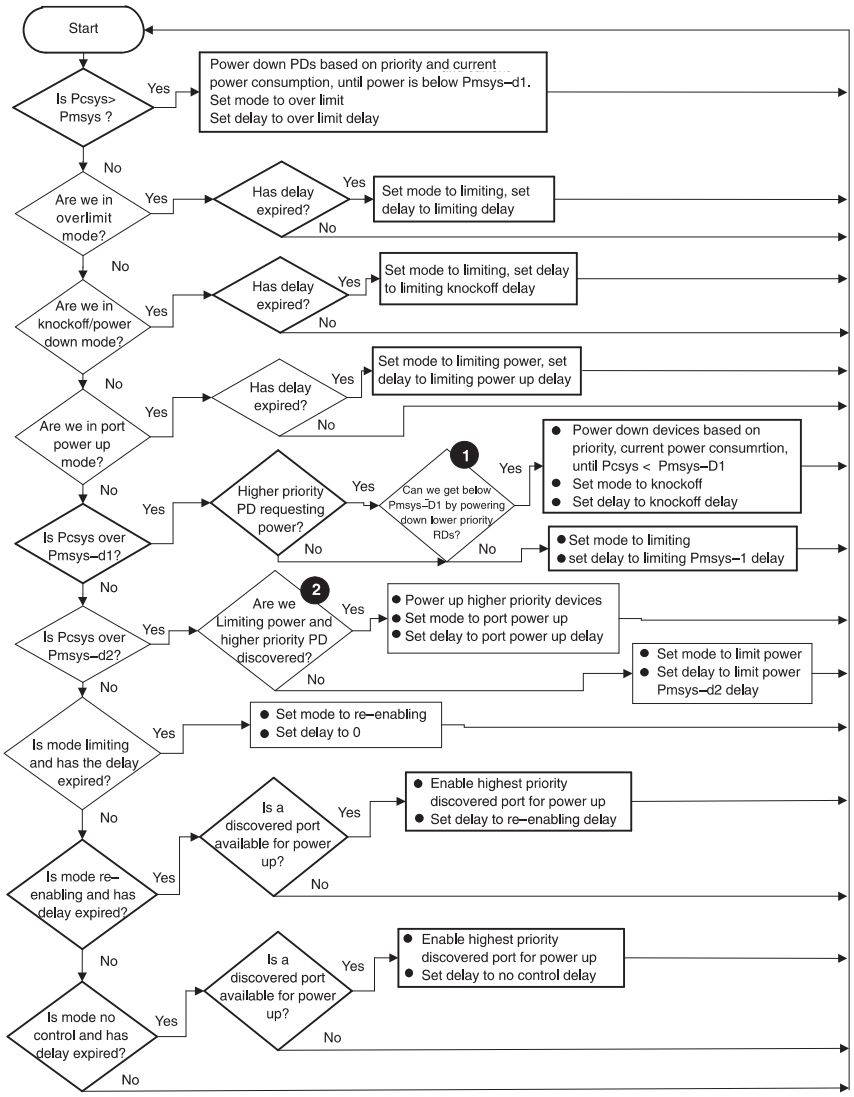

Figure 4-1. Power Management Algorithm

In Figure 4-1, there two decision points are numerically marked that require further elaboration:

1 – For this point, the test should be the following:

Is a device of higher priority discovered, fault free, and unpowered, and is a device of lower priority powered up, and can the system get below $P_{msys-d1}$ without powering down any PDs at the same priority as the unpowered device?

2 – For this point, the test should be the following:

Is the mode limiting power, and is a device of higher priority discovered, fault free and unpowered and is a device of lower priority already powered up?

4.6 Pseudo-Code

The Power Management algorithm can also be represented by the following pseudo-code.

```
if (Pcsys > Pmsys) {
         Power down PDs; /* based on priority and current power consumption
                            Until Pcsys < Pmsys-D1 */
         mode = OVERLIMIT;
         delay = OVERLIMIT _DELAY;
} else if (mode == OVERLIMIT) {
       if (delay has expired) {
       mode = LIMITING;
       delay = LIMITING_DELAY;
   }
} else if (mode == KNOCKOFF) {
    if (delay has expired) {
       mode = LIMITING;
       delay = LIMITING_KNOCKOFF_DELAY;
   }
} else if (POWER_UP) {
    if (delay has expired) {
       mode = LIMITING;
       delay = LIMITING_POWER_UP_DELAY;
   }
} else if (Pcsys > Pmsys-D1) {
    if (device of higher priority is discovered, fault free and unpowered,
    and a device of lower priority is powered up and if we can get below
    Pmsys-D1 without powering down any PDs of the priority of the
    unpowered device) {
        Power down PDs based on priority and current power consumption, until
        power is below Pmsys-D1;
        mode = KNOCKOFF;
        delay = KNOCKOFF_DELAY;
   }
   else { /* No device wanting to knock off a lower priority device */
      mode = LIMITING;
      delay = LIMITING_PMSYS_D1_DELAY; /* keep limiting until power demand drops*/
   }
} else if (Pcsys is above Pmsys-D2) {
        if (mode == LIMITING &&
           a device of higher priority is discovered, fault free, and unpowered &&
           a device of lower priority is powered up) {
           Power up higher priority device;
           mode = POWER_UP;
           delay = POWER_UP_DELAY;
        } else {
           mode = LIMITING;
           delay = LIMITING_PMSYS_D2_DELAY;
        }
} else { /* power consumption must be at or below threshold 'B' */
          if (mode == LIMITING && delay has expired) {
              mode = REENABLING;
              delay = RESET_DELAY;
         } else if (mode == REENABLING_PORTS && delay has expired) {
          if (a discovered port is not enabled for power up) {
              Re-enable highest priority discovered port for power up;
              delay = REENABLING_PORTS_DELAY;
          } else { /* all discovered ports are enabled */
               mode = NO_CONTROL_NEEDED;
          }
     } else if (mode == NO_CONTROL_NEEDED && delay has expired){
          Re-enable highest priority discovered port for power up;
          delay = ENABLE_PORTS_DELAY;
          }
     }
}
```

4.7 Power Management and Port Mapping

when deciding which ports to power up and down during the course of operation, the power management algorithm makes its decisions based on the port number and port priority.

Port Mapping binds the physical port number (which is based on physical location of the port in the system, numbered from 0 to n) to a logical port number. The logical port number is used by the Power Management algorithm when making its processing decisions. Therefore, customers who wish to alter the order in which ports are powered up or powered down can do so by mapping the physical ports to logical port numbers. Power management applies its policy decisions to the logical port numbers in sequential order, for a given port priority. See the section on *Port Mapping* for more information.

Port Mapping

Different system implementations assign the physical port numbers to microcontrollers and PSEs in a system in an arbitrary fashion. This is not necessarily the way in which user's want to prioritize ports for powering up and powering down during power management policy decisions.

Port Mapping allows the user to select a logical port number for each physical port in the system. This logical port number is used by the core firmware algorithm for policy decisions such as the order in which to power up/down ports in Power Management (for ports with the same port power priority). Port Mapping can be configured through the Host Interface Protocol by the Port Write Command. This command identifies the logical port number for the port, when configuring it.

Firmware operations such as Power Management and transactions in the Host Control Protocol operate on ports based on their logical port number. For example, when a user disables a port using the Host Control Protocol, a command to a logical port is issued. The corresponding physical port mapped to the logical port is disabled. Also, the Power Management algorithms take the logical port number into account while working with the port.

Note the following implementation details:

- By default, (or factory defaults) each physical port in the system is mapped to the same number logical port.
- The new port mapping goes into effect only after a Save/Restore Configuration message is sent, with a minimum of the System Settings being saved.
- It is the Host Processor's responsibility to verify that there is no inconsistency in the port numbering, and that the same port does not appear more than once in the matrix, etc.

The *Restore Factory Defaults* option restores the port mapping to its default configuration.

Legacy Detection

Before the IEEE certified the Power over Ethernet specification (IEEE 802.af), there were a number of vendors who had produced pre-standard Powered Devices (PDs). Because of the proprietary nature of these devices, they require different voltage and current “signatures” than standard devices in order to be detected and powered.

The goal of Legacy Detection is to identify these devices based on their unique electrical signatures (resistive and capacitive). Once identified; these Legacy devices are treated like any other PD, and can be powered-up.

The Legacy Detection feature is disabled by default, and must be enabled on a system or per-port basis through the *Host Interface Protocol* or *GUI*.

6.1 Supported Legacy Devices

The firmware for the TPS2384 supports a handful of popular Legacy Devices. These devices are:

- Cisco AP 1100 WAP
- Cisco AP 1200 WAP
- Cisco AP 350 WAP
- Cisco 7910 IP Phone
- Cisco 7940 IP Phone
- Cisco 7960 IP Phone
- Nortel i2002 IP Phone (with Power-Splitter/Dongle)
- Nortel i2004 IP Phone (with Power-Splitter/Dongle)

6.2 Legacy Device Characteristics

IEEE compliant PDs have an almost pure resistive signature of about 25 kΩ. The Legacy devices that are supported exhibit some resistive and some capacitive signatures. The TPS2384 in Auto mode only does resistive signatures. It is designed to measure capacitance under manual control, however, and the firmware takes advantage of this feature.

6.3 Signatures and Algorithm

The discovery algorithm needs to differentiate between an open port, an IEEE compliant PD, a Legacy Device, and a non PoE device. The firmware uses up to 7 readings for discovery.

- Voltage before discovery
- Discovery 1 current (defined as I_1 here)
- Discovery 2 current (defined as I_2here)
- Discovery 3 current (defined as I_3 here)
- Discovery 4 current (defined as I_4 here)
- Voltage before constant current (defined as V_1 here)
- Voltage before constant current (defined as V_2 here)

This signature technique is described below.

6.3.1 Voltage Before Discovery

Measuring any value on the input port with the TPS2384 causes a low level of leakage onto the port. This is because of the input resistors and bias resistors on the opamp inside the part.

This leakage has very different implications for different port configurations. With a standard PD resistance of 25 kΩ, it is inconsequential. With an open port, it pulls the port all the way to 48 volts. With a legacy PD that has impedance significantly higher than 25 kΩ, it is difficult to get the port down to 4.4 volts for the first resistance measurement.

The firmware has a threshold for port voltage before discovery. If the voltage is above that threshold, the discovery process is not started. This means that an open port will never have discovery voltages applied, because the port voltage will never go low enough. With a standard PD, the voltage drops below the threshold almost immediately. With some legacy devices that have some capacitance, specifically the Nortel phones with dongle, it can take 2 seconds for the voltage to drop to the threshold.

6.3.2 Discovery Currents

The TPS2384 firmware uses a 4 point discovery process for standard discovery. This involves putting out 4.4 volts, 8.8 volts, 4.4 volts, and 8.8 volts, and measuring the current each time.

Each voltage is output for about 75 milliseconds (DC-Disconnect Mode) or 50 milliseconds (AC-Disconnect Mode) at a time. This timing is dictated by real-time constraints on the system, specifically the speed of the opto-isolators on the I^2C bus connecting to the TPS2384s. The actual current measurement is made on the current during the first 18 milliseconds after the voltage is applied.

The 4 point discovery method is very useful for detecting and sometimes calculating capacitance. If there is a significant capacitor on the PD, the current for the second 4.4 volt discovery will be less than the first one. This is because the capacitor will be charged up to 8.8 volts by the preceding step, and will then be discharging during the second 4.4 volts.

In fact, the standard discovery has a correction factor for the 0.22 µF capacitor across the P and N pins of each port. That capacitor causes about a 15 µamp drop in current between Discovery 1 and Discovery 3.

With legacy PDs, it takes longer for the capacitor to charge up at both voltage levels. This can lead to two different scenarios:

1. Discovery 1 and 2 currents higher than Discovery 3 and 4
2. Discovery 2 current at maximum, Discovery 3 at minimum

With case 1, where the currents are not too high, the 4 currents represent an accurate signature of both resistance and capacitance for the legacy device. In this case, the firmware simply has windows which it applies to all 4 currents. If the PD fits within these windows, it is powered up. Refer to the software to see these windows. They are subject to change as characterization of both the TPS2384 and the legacy devices continues.

If the 4 discovery currents are sufficient for discovery, the firmware goes directly to power up, and does not use the remaining 2 measurements.

In case 2, there is not enough information yet to characterize the device. Case 2 means that the capacitor was too big to charge during the discovery time. In this case, we use another capability of the TPS2384 – the capacitance measurement, requiring two more voltage measurements.

6.3.3 Voltage Before and During Constant Current

To determine capacitance, the firmware measures the port voltage after discovery. If the voltage is below a threshold, the TPS2384 is put into constant-current mode, and the firmware reads the voltage again. This is the average voltage for about the first 18 milliseconds of the constant-current output.

The difference between these voltages is proportional to the capacitance, at least where the capacitance dominates the signature. A window is then used for this value, based on TPS2384 and legacy device characterization. These values, like the discovery current windows above, will probably change as characterization progresses.

6.4 Resistive and Capacitive Ranges

The Legacy PDs that are detected are detected in specific ranges for all four current probes during the detection algorithm. Table 6-1 shows the high and low ranges for each of these probes against the legacy device that falls into that range.

Table 6-1. A/D Ranges for Legacy PDs

Legacy PD	Resistive/Current Signature Ranges (A/D counts)	Capacitive/Voltage Signature Ranges (A/D counts)
Cisco AP 350 (WAP)	600 < I_1 < 1250 1100 < I_2 < 1500 230 < I_3< 400 110 < I_4< 1500	Not Used
Cisco AP 1100 (WAP)	600 < I_1 < 1250 1100 < I_2 < 1500 230 < I_3< 400 110 < I_4< 1500	Not Used
Cisco AP 1200 (WAP)	600 < I_1 < 1250 1100 < I_2 < 1500 230 < I_3< 400 110 < I_4< 1500	Not Used
Cisco 7910 (IP Phone)	8126 < I_1 < 10736 17439 < I_2 < 22261 7910 < I_3< 10316 17386 < I_4< 22242	Not Used
Cisco 7940 (IP Phone)	600 < I_1 < 1250 1100 < I_2 < 1500 230 < I_3< 400 110 < I_4< 1500	Not Used
Cisco 7960 (IP Phone)	1050 < I_1 < 1500 1050 < I_2 < 1500 265 < I_3< 385 1050 < I_4< 1500	Not Used
Nortel i2002 with Dongle (IP Phone)	I_2 < 21000 I_3< 424	3861 < (V_2–V_1) < 10530
Nortel i2004 with Dongle (IP Phone)	I_2 < 21000 I_3< 424	3861 < (V_2–V_1) < 10530

Software Download

7.1 Introduction

The MSP430 microcontroller is a flash-based microcontroller that allows its program memory to be updated either through hardware or through software.

The software mechanism for flash programming is implemented in the firmware using the Host Interface Protocol, and is supported through the same communications interface as the host protocol (either RS-232 over serial or I^2C). This allows flash programming without using a gang or flash tool for field upgrades. The new code can be programmed remotely using the RS-232 or the I^2C communication channels.

This in-circuit programming capability is enabled by TI's new Boot Strap Loader (BSL). The Firmware-download functionality is mostly handled by the BSL. The BSL offers a well-defined and unique communication protocol which enables the HOST to initiate and program the firmware step by step and pass the control to the freshly programmed PoE application upon a successful download validation. Alternatively, JTAG may be used for memory updating and allows recovery in case of image corruption during flash programming, etc.

7.2 The Relationship Between BSL and the PoE-Application

Availability of the Firmware download capability is based on the presence of BSL code in the Flash memory. The BSL section can be programmed using only a JTAG programmer.

The BSL is capable of programming the Flash memory – with a new PoE image - in the valid memory areas specified by the data records as explained afterwards. Any attempt to overwrite and/or erase parts of BSL will be denied. After successful programming, the PoE application code and the BSL firmware will both reside side by side in the Flash memory as two independent applications.

To start the download process, the user must send a *Program Controller* message to the PoE-application. On receipt of this command by the system, the control is transferred to the BSL, which performs and monitors the ongoing download process. At the end of download process, the system is reset. After reboot, the system runs an integrity check on the newly-loaded PoE image.

If the new image is found to be valid and intact, the BSL invokes the PoE code and passes control to the PoE application in 10-20 seconds.

If the image integrity check fails, the BSL retains control and informs the HOST with a *System Info* message, when requested.

7.3 Flash Programming Procedure

The HOST has full control of the programming procedure using a very compact set of commands sent through a communication channel.

a. Enter programming mode by sending the *Program Controller* message, when either PoE-application or BSL are in control.
b. Verify that the BSL is in control by sending *System Info* message.
c. Erase Application Flash by sending the *Clear Application* message.
d. Erase Information Flash, if required, by sending the *Clear Information* message.
e. Write the Flash by sending *Receive Data Block* messages.
f. Close the Flash by sending a *Receive Data Block* message with 0 length data.
g. Terminate the programming by sending the *Reset* message.
h. Verify PoE in control by sending *System Info* message.

7.3.1 Messages

While in the BSL, the only available messages are: Reset, System Info, Program Controller, Clear Application, Clear Information, and Receive Data Block. Any other message will be responded to with an Acknowledge Message with the appropriate error status.

The message formats are the same as those in Chapter 3. An Acknowledge Message is sent in response to the messages specific to programming the new PoE application.

7.3.2 Clear Application

This message is sent from the host processor to the MSP430 to erase the PoE application program. The MSP430 responds with an Acknowledge message after the application is erased, which takes approximately 2 seconds.

All flash memory specific to the chip (MSP430F1481 : 0x4000–0xFFFF, MSP430F169L : 0x1100–0xFFFF) is erased with the exception of the following areas (0xF800–0xFFDF and 0xFFFE–0xFFFF). These areas are required by the Boot Loader.

Table 7-1. Clear Application Message Format

Byte	Field	Values
1	Clear Application code	0xF0
2	Check code	0xAA
3	Check code	0x55
4-5	Checksum	0x01EF

7.3.3 Clear Information

This message is sent from the host processor to the MSP430 to erase the PoE configuration information area (0x1000–0x10FF). The MSP430 responds with an Acknowledge message after the information is erased, which takes less than 200 milliseconds.

Table 7-2. Clear Information Message Format

Byte	Field	Values
1	Clear Information code	0xF1
2	Check code	0xAA
3	Check code	0x55
4-5	Checksum	0x01EF0

7.3.4 Receive Data Block

This message is sent from the host processor to the MSP430 to send a data block for programming in to the Flash memory (within 0x1000–0xFFFF). Once the full message has been received and validated, the data is written to the flash memory. The response is not ready until all received data is written, or aborted due to an error. The MSP430 responds with an Acknowledge message. Any protected memory areas are ignored as long as other valid memory areas are referenced within the same message.

An empty message (containing a *Length* of 0) is required after the last data block has been received. This locks the Flash memory and allows the Boot Loader to configure parameters for validating the Flash memory after reset. The address of this *zero length* message must be different from the preceding Received Data Block or the zero length message will be ignored and the data will not be written to Flash.

If this message is received before the process is completed, the full download procedure must be started again.

Table 7-3. Receive Data Block Message Format

Byte	Field	Values
1	Receive Data Block code	0xF2
2-3	Address	Varies
4	Length (number of data bytes)	Varies (1 to 128)
5-...	Data	Varies
n-n+1	Checksum	Varies

7.4 Detailed Programming Sequence

Table 7-4 describes the download sequence.

Table 7-4. Flash Programming Sequence

PHASE	HOST -> PoE	PoE -> HOST
1. Initiate Programming	Send *Program Controller*	Ack Response
2. Verify in BSL	Send *System Info*	Verify *Code Type* in response is *BSL Running*
3. Clear Application	Send *Clear Application*	Ack Response (with up to 5 second delay)
4. Clear Information	Send *Clear Information* if required	Ack Response (with up to 1 second delay)
5. Send Data	Send *Receive Data Block* for all data	Ack Response (with up to 100 mSecond delay)
6. Reset MSP430	Send *Reset*	Ack Response
7. Verify in PoE	Send *System Info*	Verify *Code Type* in response is *PoE Running*

Module Configuration

8.1 Introduction

The MSP430 firmware for the TPS2384 can support up to 48-ports in a PoE system. This system can support different module configurations. The system can have a maximum of up to 6 modules. Modules can come in different port configurations and densities (e.g., 2x4, 2x6 and 2x8).

The MSP430 controller communicates with the TPS2384 devices in the modules by using up to three software-driven I^2C buses. This is accomplished by using general-purpose I/O pins on the MSP430 controller.

Note: Using Multiple I^2C Buses

Most PoE systems will be able to drive all modules using a single I^2C bus. The second and the third I^2C bus are usually used in conjunction with modules which have only limited internal I^2C addressing (e.g., modules that do not pin out any of the address pins from the TPS2384, so that they can be configured on the board).

In Table 8-1, the I²C address for a particular device is shown as a binary value with the format: A5-A4-A3-A2-A1 where A<n> represents the value of that address pin on the TPS2384 PSE. Note that in externally addressable modules, it is recommended that the A4 and A5 address pins be left free, for attaching multiple modules or DIMMs to the same I²C bus. Each module, DIMM, etc. contains a set of TPS2384 devices for every 4 ports (identified by letters A-F in Table 8-1).

Table 8-1. I2C Addressing Recommendations

Module Type	TPS2384 Device Address					
	A	B	C	D	E	F
2x4 module, 8 port DIMM, 2 x TPS2384	xx001	xx010	N/A	N/A	N/A	N/A
2x6 module, 12 port DIMM, 3 x TPS2384	xx001	xx010	xx011	N/A	N/A	N/A
2x8 module, 16 port DIMM, 4 x TPS2384	xx001	xx010	xx011	xx100	N/A	N/A
20 port DIMM, 5 x TPS2384 4 x TPS2384	xx001	xx010	xx011	xx100	xx101	N/A
24 port DIMM, 6 x TPS2384	xx001	xx010	xx011	xx100	xx101	xx110
2x8 early 10/100 module not externally addressable	00001	00010	0100	1000	N/A	N/A
2x4 early 10/100 module not externally addressable	00001	00010	N/A	N/A	N/A	N/A

At initialization time, the PoE-system must be configured according to the number and type of installed modules. The module configuration provides the system with the following details:

- The I²C bus (1, 2, or 3) used to communicate with the modules/TPS2384 devices.
- The I²C address of each TPS2384 device (5-bit, 3-bit internal + 2-bit external).
- The first port number on each module (Used for AC-disconnect signal toggling).
- Number of installed modules.
- The total number of ports in the system.

The above information is saved in non-volatile memory and only needs to be configured once.

8.2 Module Default Configuration

The firmware default module configuration is as follows:

- 12 port system
- 1 module
- 12 ports on each Module (2x6 module)

8.3 Module Configuration Examples

The following are examples for existing Module Configurations.

48-port system consisting of 3 X 10/100 Modules of 2X8 size.

12-port system / 1 Gigabit module of 2X6 size.

8.3.1 GUI Support

To perform Module configuration using the GUI, select the *Set Configuration* button in the *I²C Mapping* section.

A new window is opened. A new configuration may be set by manually entering data into the required fields and pressing the set button after each entry. Alternatively, the system configuration can be set to a known configuration pressing a single button in the upper side of the window.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265