CC85xx Family User's Guide

FW1.4.2

Table of Contents

1 PurePath Wireless Introduction

1.1 Device Family Overview

1.1.1 Introduction

By employing proprietary technology, referred to as PurePath Wireless, the CC85xx device family provides robust, high-quality, short-range 2.4 GHz wireless digital audio streaming in low-cost single chip solutions.

A PurePath Wireless audio network uses a star topology formed by one protocol master and up to four protocol slaves. Great care has been taken to ensure that this audio network provides gap-less and robust audio streaming in varied environments and that it can coexist amicably with existing wireless technologies in the crowded 2.4 GHz ISM band.

Most applications can be implemented without any software development and only require the CC85xx to be connected to an external audio source or sink (such as an audio codec, S/PDIF interface or class-D amplifier) and a few push buttons, switches or LED(s) for human interaction. The external audio device control is I^2C and/or GIO-based, and includes local or remote volume control. Advanced applications can interface a host processor or DSP directly to the CC85xx and directly stream audio and control most aspects of device and audio network operation.

Power management allows for timeout-based automatic power-down and power-reduction upon network inactivity and audio input/output silence, and includes also battery voltage monitoring with shut-down voltage thresholds.

The CC85x1 devices also support implementation of wireless USB soundcards with human interface device (HID) functionality, including play/volume controls, keyboard and mouse.

The PurePath Wireless Configurator, a PC-based configuration tool, is used to set up the desired functionality and parameters of the target system and then produces firmware images that subsequently must be programmed into the embedded flash memory of each CC85xx. The PurePath Wireless Configurator contains a detailed integrated help system that should be used in conjunction with this document in order to fully understand the configurable nature of the CC85xx.

All devices in the CC85xx family interface seamlessly with the CC2590 and CC2591 2.4 GHz RF range extender devices, to allow for even wider RF coverage and improved robustness in difficult environments. CC85xx protocol slaves may also control an external switch for antenna diversity to reduce RF multi-path fading effects.

1.1.2 PurePath Wireless Devices

The PurePath Wireless device family consists of four CC85xx devices. An audio network allows any mix of CC85xx devices, but the audio interface type and capabilities of a node is determined by the device according to Table 1.

Table 1 - PurePath Wireless devices

Device	Max number of audio channels	USB interface
CC8520	2	No
CC8521	2	Yes
CC8530	4	No
CC8531	4	Yes

1.1.3 Feature Set

An overview of the features of the CC85xx device family and PurePath Wireless audio networks is provided below. This list is not exhaustive; for more details see the corresponding sections in this document and the help system in the PurePath Wireless Configurator.

- Modes of operation
  - Autonomous operation. The CC85xx operates autonomously
    - The protocol master automatically establishes its network at power-up
    - Protocol slave(s) are associated with the network using push-button- and proximity-based pairing schemes, and will automatically rejoin it upon network connection loss and at later power-ups
    - Implements human interaction interface through push-buttons, switches and status LED(s) that can be connected to available GIO pins:
      - Button to start pairing operation
      - Buttons to increase/decrease volume
      - Button to toggle mute
      - Buttons to control left/right balance
      - Button to control power state of device
      - Button or switch to select audio channel(s)
      - Remote control functionality
      - Indicate network/battery status through LED blinking patterns
    - Initializes and controls, through GIO and/or I^2C, supported external audio devices (codecs, ADCs, DACs, S/PDIF receivers/transmitters, digital amplifiers, PWM processors etc.)
    - Implements its own power control, including automatic timeout-based power-down and external audio device power-reduction upon audio silence and network inactivity
      - Battery voltage monitoring for automatic shutdown and LED-based warning at specified voltage thresholds
      - Network standby mode allows for shutting down audio processing and external audio device across the entire network while maintaining network connection
  - USB device. The CC85x1 protocol master operates autonomously and is controlled by the USB host (computer or similar)
    - Automatically creates audio network upon power-up. Implements fixed/push-button pairing scheme, and a network/power status LED.
    - Implements a USB composite device consisting of
      - Audio Device for audio streaming
        - Targets applications such as USB wireless 2.0 and 2.1 speaker systems, headphones (i.e. without microphone), headsets (i.e. with microphone) and microphones
        - The USB host controls volume and muting on the protocol slaves through remote volume control

- Human Interface Device (HID) for remote control functionality (from protocol slaves), using either pre-defined or custom-defined reports:
  - Pre-defined:
    - Play control, supporting commands for controlling media player applications and similar (play/pause, skip, volume control etc.)
    - A full 101-key standard US keyboard
    - Two-axis mouse with up to 8 buttons (including left/middle/right)
  - Custom-defined:
    - Up to four HID reports consisting of up to 12 bytes of data in total (on/off and absolute value type controls only)
    - HID-specific USB descriptors and HID report formats specified by user-written XML-based definition file.
    - Report contents generated directly by one protocol slave

- Complies with the following USB standard definitions:
  - USB Specification 2.0
  - Audio Device Class Definition 1.0
    - Optional compliance with Audio Device Class Definition for Basic Audio Devices 1.0
  - Audio Data Formats 1.0
  - Audio Terminal Types 1.0
  - HID Device Class Definition 1.11
  - HID Usage Tables 1.11
- Implements USB suspend. The CC85x1 will power down automatically upon USB bus inactivity and resume operation when bus activity is resumed.
- Implementation maximizes compatibility with different operating systems, with no need for custom USB driver development. No technical knowledge of the USB standard is required to develop a product.

- Host-controlled operation. Various aspects of the CC85xx's operation are controlled by a host processor (typically MCU or DSP) through a set of SPI commands. Allows greater flexibility than for autonomous operation
  - Host processor is responsible for establishing the networks for protocol masters, and scanning for/pairing to and (re)joining networks for protocol slaves
  - Host processor must control the local volume but is informed of changes to remote volume and can, on the protocol master, set remote volume
  - Host processor is responsible for controlling the power state of the CC85xx and possibly external audio devices. It can poll information like audio silence, network inactivity and battery voltage that could affect the system power state.
    - Network standby mode allows for shutting down audio processing and external audio device across the entire network while maintaining network connection.
  - Host processor on a protocol slave can dynamically control which audio channels to receive and stream. On a protocol master the audio channel selection is statically defined at configuration time.
  - Host processor must implement the desired human control interface and status reporting, but can make use of CC85xx's status LED as in autonomous operation
  - Host processor can implement initialization and control of the external audio device, or let the CC85xx control the device as in autonomous operation
  - Host processor can utilize a low-bandwidth integrity-checked data side-channel

- Wireless Audio Networking
  - Star network with one *protocol master* device and up to four *protocol slave* devices. The protocol master:
    - Controls all aspects of the wireless audio streaming protocol, protocol slaves are only allowed to transmit when explicitly given consent
    - Defines which of 16 pre-defined logical audio channels are used in the audio network
    - Is the source of the audio clock used by all nodes in the system
  - Wireless audio streaming protocol
    - Timeslotted: The protocol master divides time into fixed time intervals (nominally 2.5 ms) during which each node in the audio network communicates on the same RF channels
      - With timeslot alignment, multiple protocol masters can be used in close proximity to support applications that require more protocol slaves and/or audio channels
    - Adaptive frequency-hopping: 18 RF channels with a 4-MHz bandwidth are defined. The protocol master adaptively decides which subset of RF channels to use at any given instance based on historical performance (interference from other radio systems, multipath fading, etc).
      - Manual frequency planning with a minimum of 6 RF channels is possible at configuration time or through the external host interface
    - Carrier sense for improved co-existence: The protocol master measures energy on each RF channel in a listen-before-talk operation and then decides whether to transmit on the RF channel or not
    - Seamlessly interfaces CC2590 and CC2591 range extender devices for improved range
    - Supports antenna diversity by using an external switch with two antennas
  - Wireless audio streaming
    - All audio channels in an audio network use the same sample rate
    - Each audio channel used in an audio network is assigned to a logical channel (e.g. front left) to facilitate a common understanding
    - Audio can be streamed from protocol master to protocol slaves or the opposite direction, as determined by the logical channel identifier
    - Audio channels produced by the protocol master can be consumed by up to 4 protocol slaves
    - In autonomous operation, some types of protocol slaves can implement dynamic channel selection, controlled by push-button, by switches or automatic selection. In host-controlled operation, the protocol slave's channel selection is always dynamic.
    - Each audio channel has a configurable audio streaming format for a robustness/channel count/quality trade-off:
      - PCM16: Uncompressed CD-quality 16-bit PCM data [706 kbps/ch @ 44.1 kHz]
      - PCMLF: Low-frequency 16-bit PCM channel where a high-quality decimation-by-four is performed at the audio source and an interpolation-by-four is performed at the audio sink [176 kbps/ch @ 44.1 kHz]
      - SLAC: A proprietary low-complexity, low-loss compression algorithm supplying near CD-quality [232 kbps/ch @ 44.1 kHz]
      - PCME24: Companded 24-bit PCM data, offering 15-bit signal-to-noise ratio and full 24-bit dynamic range [706 kbps/ch @ 44.1 kHz]
    - Each audio channel data stream is sliced up into small pieces that are individually transmitted and integrity checked. Pieces that are not received correctly by any audio consumer are retransmitted by the audio producer.

- PCM16 and PCME24 streaming formats implement error concealment mechanisms at the audio consumer, so that no audible break in the audio stream can be heard for marginal RF conditions
- In case the required minimum data throughput to allow audio streaming is not possible, the audio output on an audio consumer is muted on all channels on node(s) failing to receive. Hysteresis is applied to avoid situations where audio is streamed only intermittently.

- Wireless audio clock distribution
  - Master audio clock is the audio clock seen by the protocol master and it is wirelessly distributed to all protocol slaves
  - Master audio clock is either generated by CC85xx (fixed frequency) or is an input from an external audio device. A predefined set of sample rates is supported with a tolerance of ±2000 ppm: 32, 40.275 (not USB devices), 44.1 and 48 kHz. A change in sample rate is handled gracefully.
  - For bidirectional audio streaming, the same sample rate is used for both streaming directions. For USB devices supporting bidirectional streaming, only one sample rate can be used, and this rate is selected at configuration time.
  - Audio clock is synchronized to within ±1 sample across all nodes in audio network. Pin-to-pin audio latency is configurable to between 512 and 2048 samples (10.7-42.6 ms @ 48 kHz sample rate, 11.6-46.4 ms @ 44.1 kHz sample rate). See restrictions in Table 11.
  - Protocol slaves will generate audio clock from power-up to power-down except during short periods when switching sample rate. This allows for driving external circuitry that also depends on the CC85xx generated audio clock.
  - If the protocol master uses external audio clock source, and this clock source stops or generates an invalid frequency, the protocol slaves continue generating a free-running clock at the last seen valid sample rate
- Volume control
  - The protocol master distributes separate master volume settings for all output and input audio channels in the audio network, and supports also (for host-controlled and USB-based protocol masters) per-channel volume offsets
  - Each protocol slave can offset or override the master volume locally, and supports also (for host-controlled protocol slave) per-channel volume offsets
  - For autonomous operation with power control button, the last volume settings can be saved to the embedded flash memory and thus survive across power cycling

- Remote control
  - Protocol slaves (producer):
    - Autonomous operation: Can generate pre-defined remote control commands from up to 8 configurable button event(s)
    - Host controlled operation: The host processor can either generate pre-defined or custom-defined remote control input:
      - One or more protocol slaves can generate pre-defined remote control commands, keyboard codes (including modifier keys) for a standard 101-key US keyboard and movement and buttons for a two-axis mouse. The keyboard codes are defined by the USB HID Usage Tables 1.11 definition.
      - One protocol slave can generate raw data for custom-defined USB HID reports or other proprietary remote control functionality
  - Protocol masters (consumer):
    - Autonomous: Pre-defined remote control commands can be mapped to GIO outputs for wireless button functionality. Volume and network standby commands are automatically forwarded to the network's protocol slaves.
    - USB device: Can receive either pre-defined or custom-defined remote control input:
      - Pre-defined remote control commands, keyboard codes and mouse movement/buttons are converted into HID reports and transferred to the USB host. Network standby commands are automatically forwarded to the network's protocol slaves
      - Custom-defined HID reports are generated based on remote control input received from a single protocol slave
    - Host-controlled operation: The host-processor polls remote control data (codes for currently pressed keys and mouse movement, or proprietary remote control input) from each connected slave
  - Pre-defined remote control input from multiple protocol slaves is or'ed/added together
  - Timeout mechanism prevents stuck keys when protocol slaves (temporarily) leave the network
- Data side-channel
  - For host-controlled operation, a data side-channel is made available between each protocol master and slave node that can be used for user-defined purposes
  - Implements a sequenced, assured delivery datagram service in each direction. The best-case throughput is 100 kbps into and out of the protocol master.
- Finding/joining audio networks and device filtering
  - Each CC85xx has a globally unique 32-bit d*evice ID*. The device ID of the protocol master in an audio network is also the *network ID.*
  - For autonomous operation, pairing is performed using a simple scheme where a button is pushed simultaneously on both protocol master and protocol slave, or where a button is pushed on the protocol slave and a proximity criteria must be fulfilled. The latter scheme allows for button-less protocol masters. Once paired with a protocol master a protocol slave saves the network ID in its embedded flash and will automatically attempt to rejoin the same network in the future. Pre-pairing as part of production procedure is also possible, thereby avoiding need for a pairing button or for the end-user to perform initial pairing.
  - For host-controlled operation, commands for scanning for available audio networks and joining specific audio networks are available. More advanced pairing mechanisms can thus be implemented.

        - Each node has a user-defined manufacturer ID and product ID. Filtering criteria can be configured to ensure that a node will only be allowed to join a network if its manufacturer ID matches and/or its product ID passes certain criteria.
- Audio Interfaces
    - Serial audio interface
        - Interface formats supported: I²S, LJF, RJF, DSP
        - Sample resolution: 16-bit, 24-bit
        - Supported sample rates: 32, 40.275, 44.1, 48 kHz (±2000 ppm tolerance for input clock)
        - Audio clock source:
            - Internally generated (all clock/framing signals are outputs)
            - External (all clock/framing signals are inputs). Sample rate changes and start/stop of clock is handled gracefully.
        - 1-3 audio data pins allows simultaneous input/output and up to 4 audio channels
    - USB audio device (protocol masters only)
        - Sample resolution: 16-bit, 24-bit
        - Supported sample rates: 32, 44.1, 48 kHz
            - Only one sample rate can be selected at configuration time when supporting bidirectional audio streaming
        - Flexible specification of audio device topology, using XML-based USB descriptor definition files
        - Complies with the following USB standard definitions:
            - Audio Device Class Definition 1.0
                - Optional compliance with Audio Device Class Definition for Basic Audio Devices 1.0
            - Audio Data Formats 1.0
            - Audio Terminal Types 1.0
- External audio device support
    - Initialization and control of external audio devices through GIO and I^2C (e.g. volume, sample rate, low power states)
    - Support for a range of TI codecs, DACs and ADCs (including AIC3101 and derivatives, AIC3204 and derivatives, DAC32, ADC3101, AIC34)
    - Support for a range of TI digital-input class-D amplifiers and PWM processors (including TAS5708 and derivatives, and TAS5518)
    - Support for DSPs and S/PDIF-transmitters and -receivers (including DIR9001 and DIT4096), using linear PCM
    - Users may, using an XML-based file format, add support for additional audio devices that are controlled by I^2C and/or up to 4 GIO pins
    - In host-controlled operation, host processor receives notification about changes in power state, sample rate and volume to allow it to control any external audio device
- Configuration
    - CC85xx devices are delivered in a blank state, i.e. without any firmware programmed, due to the highly configurable nature of the device
    - Configuration of all CC85xx nodes in an audio network (network role, operational mode, audio configuration, pin configuration and many other parameters) is done through the PurePath Wireless Configurator tool

- PurePath Wireless Configurator generates firmware images for each node type in a network and allows, in a development phase, direct programming of CC85xx devices through the included CCDebugger USB dongle
- In production of end products, each CC85xx device must be programmed with the relevant firmware image through a set of SPI commands described in this document
- PurePath Wireless Configurator can, for any device configuration, produce a tailored production test firmware image that implements various test commands for testing I/O, RF signal path and audio signal path

1.1.4 Firmware Version Interoperability

The PurePath Wireless firmware version numbering scheme is:

FW *X*.*Y*.*Z*

where *X* is the major revision number, *Y* is the minor revision number and Z is the maintenance/patch revision number.

For greatest performance it is recommended that all nodes in an audio network use the same firmware revision. However, an audio network supports having nodes running different firmware version within the same minor revision (e.g. FW 1.0.3 and FW 1.0.4) with full functionality[1].

New FW revisions will be able to stream audio from older revisions, providing the old revision supports the required features. (For example - SLAC is implemented as of FW 1.1.0 and therefore a FW 1.0.x protocol slave will not be able to stream SLAC audio from a FW 1.1.x protocol master. The FW 1.0.x protocol slave *will* be able to stream PCM16 and PCMLF audio formats from a FW 1.1.x protocol master.)

Basic audio network functionality, such as network acquisition and joining, audio streaming and data side-channel communication, will be possible between nodes with the same major revision number (e.g. FW 1.0.3 and FW 1.1.0). This will allow, for instance, the implementation of over-the-air firmware upgrades in advanced systems.

Nodes with different major revisions (e.g. FW 1.0.3 and FW 2.0.0) may not allow any form of interoperation.

For a detailed and specific overview of version interoperability, refer to the PurePath Wireless Configurator's change log.

[1] Firmware 1.0.3 and later firmware versions are not compatible with firmware version 1.0.2 and earlier. Firmware 1.0.3 introduces improved RF packet integrity checking, which improves the robustness of the PurePath Wireless platform significantly. Refer to the PurePath Wireless Configurator change log for further details.

1.2 Application Overview

1.2.1 Concept

An audio network in a given application will exist of:

- One protocol master node
- One or more protocol slave nodes

In many applications (wireless headphones, wireless speakers or wireless audio cable replacement) it is natural that the protocol master is at the audio source, or is the audio producer in PurePath Wireless parlance. However, in some applications (wireless microphones for example) it is natural that the audio producer is one or more protocol slaves and that the protocol master is an audio consumer.

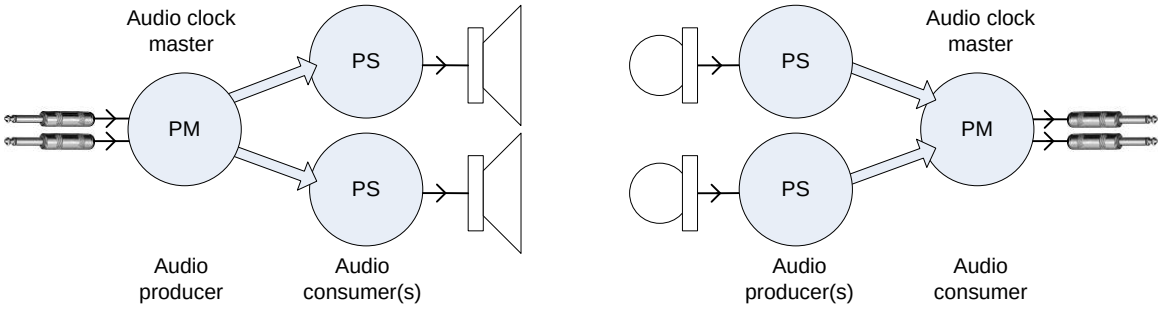


Figure 1 - Illustration of different audio network topologies

In both of the above cases, an important concept to understand is that the audio clock used by the audio network is the one generated by or input at the protocol master, the so-called audio master clock. The audio interface of protocol slaves always output a copy of the audio master clock on its audio clock and framing signals.

The configuration of the protocol master statically defines a number of parameters for the audio network:

- Radio protocol timing and maximum number of protocol slaves that can participate
- The audio sample rate if generated locally by protocol master
- The pin-to-pin audio latency in number of samples (individually for each sample rate)
- The set of audio channels (as identified by logical channel ID numbers) that the audio network supports and the streaming format used for each audio channel
- Whether the data side-channel can be supported by audio network (host-controlled operation)
- Optionally, by demanding a certain manufacturer ID (see section 3.4.1), determine which protocol slaves are allowed to join audio network

Protocol slaves can join any audio network and will adapt to the protocol master configuration. Static definitions for a protocol slave include:

- Optional filtering criteria for which audio network it will attempt to join. The filtering criteria consist of a manufacturer ID, product ID and a product ID mask (see section 3.4.1).
- Audio channel mapping, i.e. which logical audio channels in the audio network they consume/produce and how they should map to the channels of their audio serial interface. If dynamic audio channel selection is activated this can change during operation.
- Whether the data side-channel is supported (host-controlled operation)

To summarize, a protocol slave is compatible with and is able to stream audio in the audio network formed by a protocol master if:

- Manufacturer ID filtering is disabled on both protocol master and protocol slave or the manufacturer ID is an exact match
- The product ID of the protocol master passes the product ID filter criteria of the protocol slave
- The protocol slave can map at least one of the logical audio channels supported by the protocol master
- If a data-side channel is desired for host-controlled operation, this must be supported by both protocol master and protocol slave

It is perfectly permissible that some nodes in an audio network use autonomous operation while others use host-controlled operation.

1.2.2 Applications

Different types of applications have been gradually introduced through five main firmware releases:

The initial FW release (FW 1.0) targets, but is not limited to, the following applications:

- Audio cable replacement (CC8520/CC8520)
- Wireless headphones (CC8520/CC8520)
- Wireless subwoofer (CC8520/CC8520)

The second FW release (FW 1.1) adds support for multiple protocol slave nodes, more audio channels and basic power management. This enables, among other, the following new applications:

- Wireless speakers (CC8520/CC8520)
- Wireless 2.1 speaker system (CC8530/C8520)
- Four channel systems, e.g. base station with dual stereo inputs and corresponding headphones which may select dynamically between the two stereo inputs (CC8530/CC8520)

The third FW release (FW 1.2) adds support for bidirectional audio streaming, and also adds USB audio and HID device class support. This enables, among other, the following new applications:

- Wireless USB headphones (CC8521/CC8520)
- Wireless (USB) headset (CC8530/CC8531)
- Wireless USB speakers (CC8531/C8520)

The fourth FW release (FW 1.3) adds support for streaming audio from protocol slaves to protocol masters, advanced power management and antenna diversity. This enables, among other, the following new applications:

- Wireless (USB) microphones (CC8520/CC8521)

The fifth FW release (FW 1.4) adds support for protocol master timeslot synchronization, frequency planning and proprietary remote control/custom-defined USB HID reports. This enables, among other, the following new applications:

- Multi-protocol master systems, e.g. base stations containing multiple CC8530s to serve more protocol slaves and/or audio channels (CC8530/CC8520)
- Remote controls with audio capabilities

1.3 Device Operation

A CC85xx device can be configured for one of two modes of operation depending on the requirements of the application.

- Autonomous operation, including USB devices (CC85xx controls application)
  - + Very quick time to market as no software development is required
  - + No MCU required ➔ Reduced BOM and PCB area
  - ÷ Can only use pin- and/or I²C-controlled external audio devices
  - ÷ Limited options for human user interface customization
- Host-controlled operation (CC85xx controlled by host processor)
  - + Allows more flexibility for audio network operation and human user interface
  - + Data side-channel allows additional functionality
  - + More flexible use of any external audio devices if controlled by host processor - or the same external audio devices as for autonomous operation if controlled by CC85xx
  - ÷ Requires host processor and software development ➔ increased cost and development time

1.3.1 Autonomous Operation

When CC85xx is operating autonomously it handles all aspects of the application:

- Audio networking
  - Automatic network establishment for protocol masters
  - Pairing mechanism for protocol masters and protocol slaves
  - After successful pairing, a protocol slave saves network ID to its embedded flash memory and reconnects to audio network automatically in the future
- Human user interface
  - Button for starting pairing operation (not required on protocol master when protocol slave uses proximity based pairing)
  - Buttons for manipulating volume and balance (protocol slaves only)
  - Button to control power state of device
  - Protocol slave: Buttons for generating remote control commands
  - Protocol master: Remote control commands can be mapped to GIO outputs for wireless button functionality. Received remote volume control and network standby commands automatically generate the same behavior as pushing corresponding buttons on the protocol master.
  - Switch(es) or button to select which audio channel(s) to stream. Automatic channel selection can be used for mono microphones.
  - Network/battery status indicated by LED(s)
- Control of external audio circuits through GIO and I^2C
  - Safe initialization and power-down of supported devices
  - Smooth volume changes and safe sample rate changes
  - Power optimization of audio path depending on device power state
  - Audio input valid for digital inputs can be used to avoid streaming S/PDIF bitstreams

- Power control
  - If a protocol slave's local audio inputs and/or outputs are silent, it can automatically reduce audio device power consumption and/or power down after configurable timeouts
  - If no audio network is found or a protocol slave loses connection to the protocol master, it can automatically reduce audio device power consumption and/or power down after configurable timeouts
  - If a protocol master has no connected protocol slaves, it can automatically reduce audio device power consumption after a configurable timeout
  - Power control button can transition device to and wake device from lowest power state
  - Network standby button can disable audio processing and external audio device across the entire network

A typical application block diagram for a system implementing a basic wireless headphone is given below.

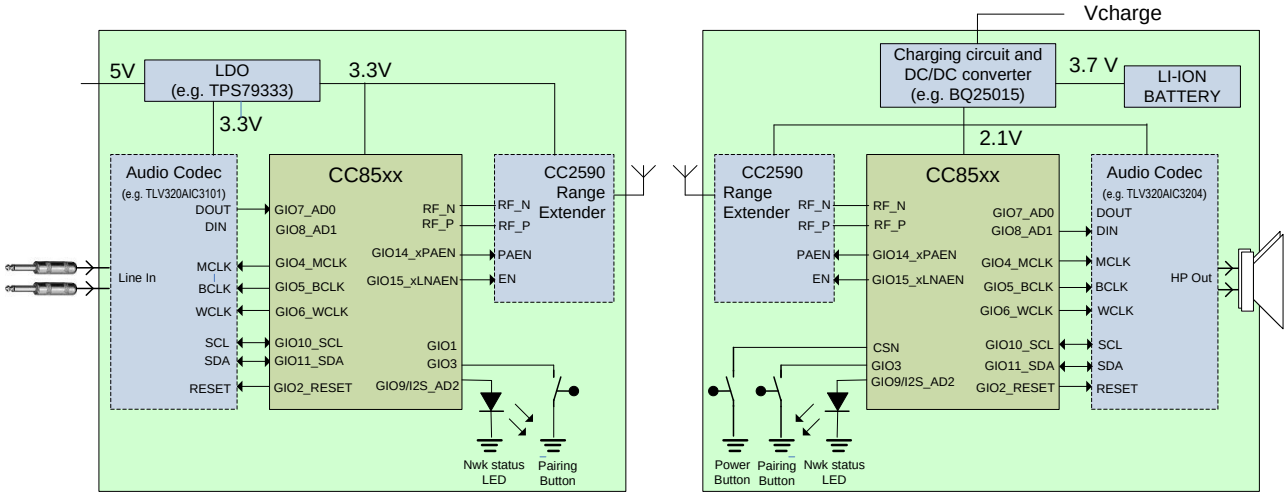


Figure 2 - Application block diagram for basic wireless headphone

1.3.2 USB Device

The CC85x1 operates autonomously, and is controlled by the USB host (computer or similar):

- Audio networking
  - Automatic network establishment (USB devices are always protocol master)
  - Pairing mechanism
- Human user interface
  - Button for starting pairing operation (not required when protocol slave uses proximity based pairing)
  - Network state and/or power state status indicated by LED
  - HID functionality via remote control command receiver allows protocol slaves to control certain aspects of the USB host (e.g. volume and media player control)
- Power control:
  - The USB host controls the power state of the USB device. The CC85x1 powers down with state retention upon missing USB bus activity (USB SUSPEND), and resumes operation when USB bus activity continues (USB RESUME).
  - Received network standby remote control commands are processed in the same way as for autonomous protocol masters
- The USB host controls the audio streaming
  - Controls volume, muting and sample rate (unless fixed)
  - Starts and stops transfer of audio samples

A typical application block diagram for the protocol master side of a system implementing a USB sound card is given below.

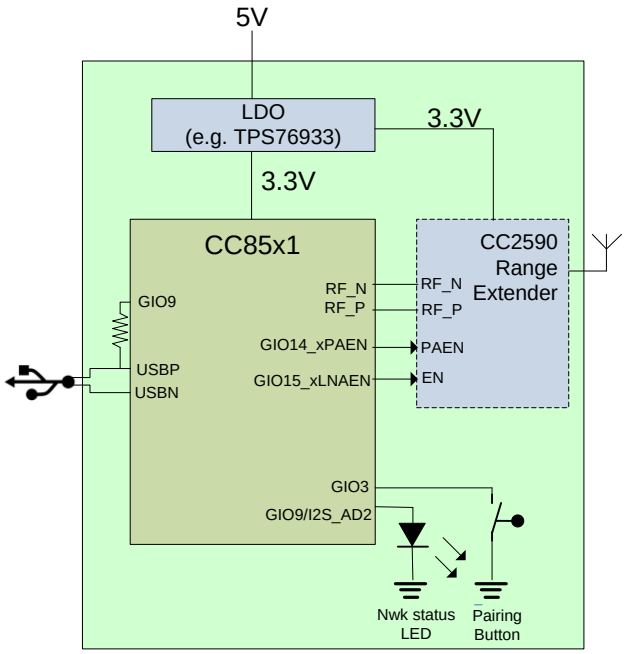


Figure 3 - Application block diagram for USB device

1.3.3 Host-Controlled Operation

In host-controlled operation a host processor is responsible for some aspects of the CC85xx's operation that are handled automatically in autonomous operation:

- Audio networking
    - On the protocol master, the host processor must manually enable and disable network maintenance and control pairing signaling
    - On protocol slaves, the host processor must manually
        - Scan for networks, for instance to implement button- or proximity-based pairing schemes as in autonomous operation. Host processor is responsible for any non-volatile storage of network ID[2].
        - Join and re-join the desired network at startup and whenever network connection is lost
        - Select which logical audio channels to subscribe to after joining
- Human user interface
    - Host processor must implement a human user interface for pairing, volume manipulation, audio channel selection and power control.
    - Host processor can implement status reporting or use the same network/power state LED indicator as for autonomous operation
    - Pre-defined remote control commands/keyboard key states/mouse movement/mouse buttons or proprietary remote control input/custom-defined HID report data may be transmitted from protocol slaves, and be polled on the protocol master. This is compatible with the remote control functionality implemented by autonomous operation and USB devices.
- Control of external audio circuits
    - CC85xx can use one of the external audio device drivers used for autonomous operation or the host processor can implement initialization/control of external audio circuit by itself
    - In the latter case, the external host interface provides information about events that the host processor needs to know about such as sample rate changes, volume changes and network events
- Power control
    - The host processor is responsible for controlling the power state of the CC85xx in response to audio network events, reported power management related information and user input

Host-controlled operation gives an application:

- More flexible control of audio network scanning, joining and pairing
- More freedom in implementing the desired human user interface
- More flexible control of volume, which audio channels to stream and choice of external audio devices
- Access to data side-channel opens up for more advanced applications (including such as over-the-air firmware upgrade)

A typical application block diagram for the headphone side of a wireless headphone application using host-controlled operation is given below.

[2] The NVS_SET_DATA and NVS_GET_DATA commands can be used to store information on the CC85xx

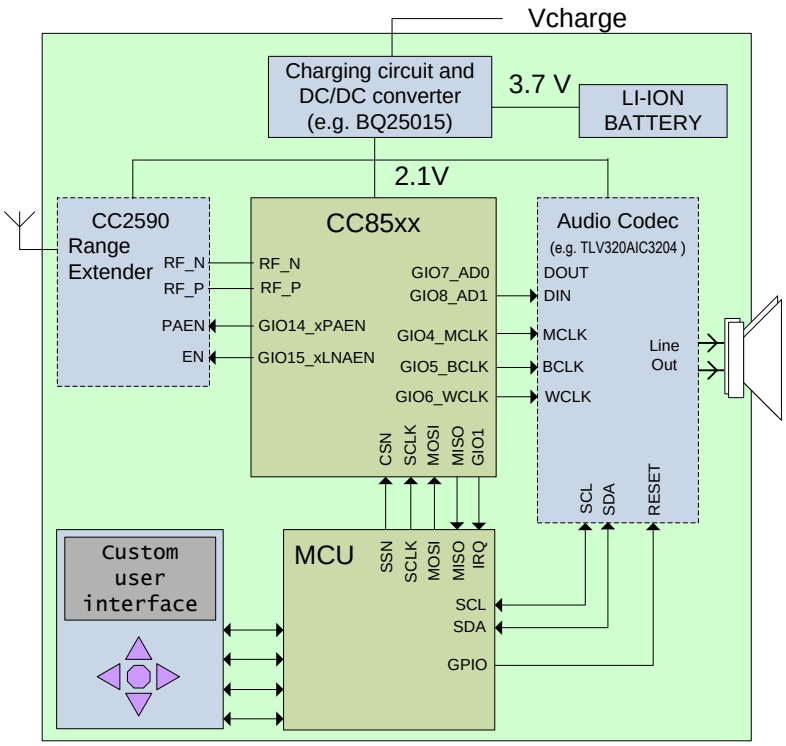


Figure 4 - Application block diagram for wireless headphone with host-controlled operation

1.3.3.1 Host Processor Hardware Requirements Summarized

The host processor must have the following hardware features to be able to operate the CC85xx:

- A 4-wire SPI interface used to operate the external host interface
- Two optional GIO pins for external host interface interrupt and CC85xx reset pin control

The CC85xx provides non-volatile storage for four 32-bit values. The host processor can read and write these values over the EHIF interface. If four 32-bit values is not sufficient for non-volatile storage of pairing information, audio volume settings and other variables during system power-down, the host processor needs access to internal or external software writable flash memory, EEPROM or similar for this purpose.

1.4 Device Configuration

The CC85xx device family is based on firmware running from on-chip flash memory. This firmware defines the various properties of a CC85xx device, such as network and application role, how audio is streamed, how the device is operated and how CC85xx fits into the application circuit.

Since all aspects of the device behavior are defined by firmware, there is a need for different firmware images for different applications. The CC85xx devices are therefore delivered unprogrammed from Texas Instruments.

The firmware is generated by a PC-tool with a graphical user interface called the **PurePath Wireless Configurator**. The configuration process is based purely on settings customization, and does not require any software development or any third-party tools. When all settings have been made, the Configurator generates the firmware image, and can download it directly into CC85xx through the CC Debugger (for evaluation) or output it to an Intel HEX-file (for production programming). Refer to appendix B.2 for a detailed description of the CC85xx boot sequence and flash programming.

The image generation process is based on patching pre-compiled firmware images from the Configurator's image database.

1.4.1 PurePath Wireless Configurator

The Configurator operates on device configurations contained in projects. There is one device configuration for each different firmware image to be produced.

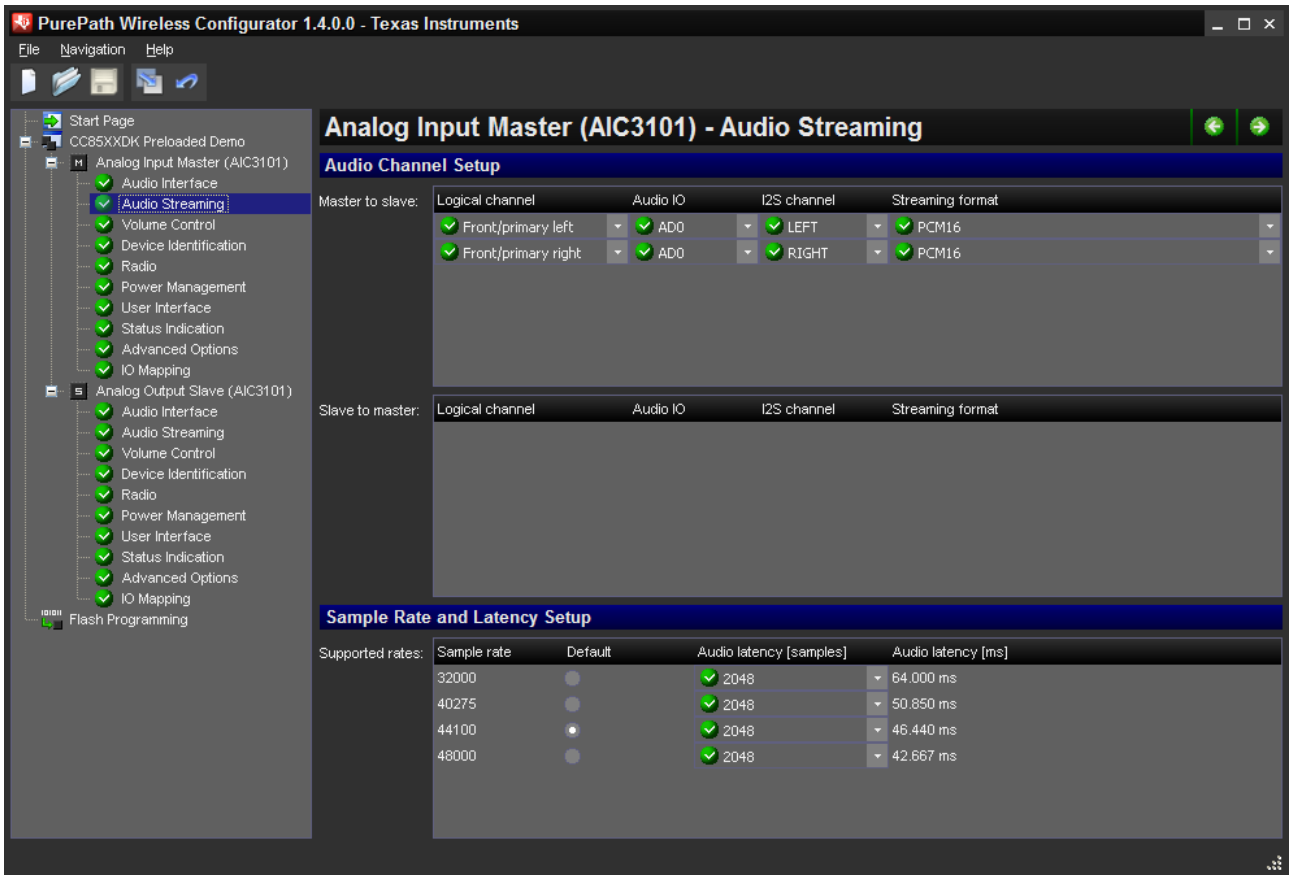


Figure 5 - PurePath Wireless Configurator screenshot

Several example projects are shipped with the PurePath Wireless Configurator, providing a quick starting point.

Below is an overview of the major configurable device settings. Note that some settings may not be available, depending on values of other settings; for instance, if host-controlled operation is selected there will not be any button-based user interface. For further information, access the Configurator's integrated help system.

Device Configuration:

- Network and application role
- Operational mode (autonomous or host-controlled)
- Hardware platform
- USB descriptor template selection for USB devices
- **Audio Interface:**
  - External audio device (e.g. TLV320AIC3101 audio codec)
  - External audio interface clock and data format
- **Audio Streaming:**
  - Mapping between channels on the local audio interface (connected to the external audio device) and the audio streams transferred over the PurePath Wireless network
  - Audio streaming format for each individual audio stream
  - Sample rates and audio latency
- **Audio Device Customization:**
  - For the selected external audio device: Custom configuration of register settings, GIO pin control and pin monitoring (e.g. input audio valid pin)
- **Volume Control:**
  - Routing of volume control information
  - Volume and balance control buttons
  - Minimum and maximum volume etc.
- **Device Identification:**
  - Manufacturer and product identification
  - USB device identification
- **Radio:**
  - Network pairing options (button or fixed), including filtering on manufacturer and product ID
  - RF parameters (e.g. target TX output power) and frequency planning
  - Maximum number of protocol slaves allowed by protocol master
- **Power Management:**
  - Power/network standby toggle button
  - Timeout constants for automatic power-down and power reduction
  - Silence detection thresholds for inputs
  - Battery voltage monitoring
- **User Interface**
  - Button timing
  - Remote control command selection for autonomous operation button input/GIO output
- **Human Interface Device**
  - USB human interface device (HID) configuration and report format overview
- **Status Indication:**
  - Network and battery status LED(s) (blink patterns).

- **Advanced Options:**
  - Advanced settings for optimizing network and audio streaming performance
  - Parameters in non-volatile storage during power-down
- **IO Mapping:**
  - Mapping of IO functions (e.g. network pairing button) to IO pins (e.g. GIO3)

2 System Overview

2.1 Hardware Overview

This section provides a brief overview of the CC85xx hardware.

2.1.1 Block Diagram

The block diagram in Figure 6 shows an overview of the hardware building blocks that a CC85xx device consists of. The modules can be roughly divided into six categories: Digital peripheral modules, CPU cores, memory, power supply, clock generation and the radio transceiver.

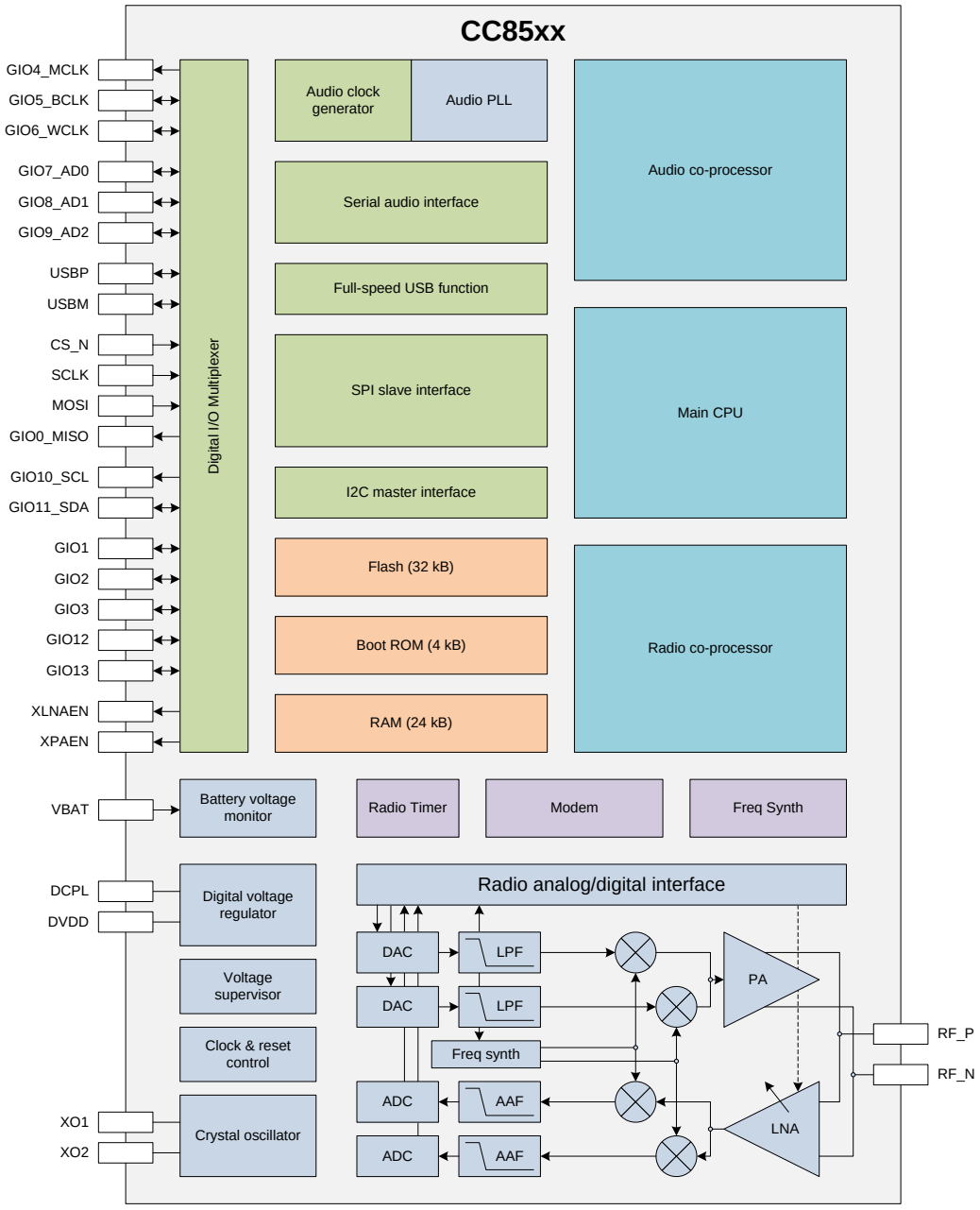


Figure 6 - Block diagram

2.1.1.1 CPU Cores

There are three on-board CPU cores:

- Audio co-processor: Proprietary CPU core used to convert between the sample format used by the serial audio interface/USB audio device and the streaming format used by the radio
- Radio co-processor: Proprietary CPU core used to control the radio and process incoming and outgoing packets
- Main processor: ARM Cortex-M3 CPU core used to coordinate co-processors and peripheral modules and perform general management functions

2.1.1.2 On-Chip Memory

On-chip memory consists of 4 kB of ROM, 32 kB of flash memory and 24 kB of RAM. The flash memory is programmed using a ROM-based bootloader. The boot sequence is described in detail in appendix B.2.

2.1.1.3 Peripheral Modules

The CC85xx features the following peripheral interfaces:

- Serial audio interface
- Full-speed USB interface (CC85x1 devices only/protocol masters only)
- SPI slave interface
- I^2C master interface
- CC259x range extender control interface
- Battery voltage monitoring
- Antenna switch control interface (protocol slaves only)

2.1.1.4 Radio Transceiver

The radio transceiver in CC85xx has:

- Differential antenna port
- An integrated frequency synthesizer with 1 MHz step size for any frequency in the range 2400-2483 MHz
- Complex 3 MHz IF TX signal chain with programmable gain PA
- Complex zero-IF RX signal chain with 36 dB variable gain in the LNAs and 70 dB dynamic range ADCs
- Proprietary baseband processing with shaped-8FSK modulation at a 2 MHz symbol rate using a rate-5/6 4D trellis coded modulation scheme to achieve a 5 Mbps raw data rate, or shaped-2FSK modulation to achieve 2 Mbps raw data rate

2.1.2 I/O Mapping

There are in total 21 digital I/O pins, which can be used either in peripheral mode or in general-purpose I/O mode:

- IO functions based on peripheral I/O (e.g. the I^2C interface) are bound to specific pins and cannot be moved
- IO functions based on general purpose I/O (e.g. button input) can be assigned random GIOx pins. The mapping is done at configuration time using the PurePath Wireless Configurator.

Below is an overview of all digital I/O pins.

Table 2 - Digital I/O pin overview

Pin #	GIO	Peripheral	Peripheral type	Peripheral function
1		USBN XANTN	USB interface Antenna switch control	CC85x1 protocol master: USB D- data line CC85xx protocol slave: Antenna switch control
2		USBP XANTP	USB interface Antenna switch control	CC85x1 protocol master: USB D+ data line CC85xx protocol slave: Antenna switch control
3		CSn	SPI slave interface	SPI active low chip select
4		SCLK	SPI slave interface	SPI serial bit clock
5		MOSI	SPI slave interface	SPI master data output
6	GIO0*	MISO	SPI slave interface	SPI master data input
7	GIO1			
8	GIO2			
9	GIO3			
13	GIO4	MCLK	Serial audio interface	Audio device master clock
14	GIO5	BCLK	Serial audio interface	Audio interface bit clock
15	GIO6	WCLK	Serial audio interface	Audio interface word clock
16	GIO7	AD0	Serial audio interface	Audio interface data line 0
17	GIO8	AD1	Serial audio interface	Audio interface data line 1
19	GIO9	AD2	Serial audio interface	Audio interface data line 2
32	GIO10	SCL	I^2C master interface	I^2C serial bit clock
33	GIO11	SDA	I^2C master interface	I^2C serial data line
34	GIO12			
35	GIO13			
36	GIO14	XPAEN	Range extender control	PA enable
38	GIO15	XLNAEN	Range extender control	LNA enable

* Note that GIO0 behavior is changed when the CSn pin is low, and is therefore not configurable for general-purpose I/O usage.

2.1.3 SPI Slave Interface

CC85xx features a standard 4-wire slave serial peripheral interface (SPI). By using the basic SPI operations defined in appendix B.1, an external SPI master can:

- Perform flash programming through the CC85xx bootloader. See appendix B.2 for details on the CC85xx boot sequence and the command set used for programming and verification.
- In host-controlled operation, operate the CC85xx through the external host interface (EHIF) command set. See appendix B.3 for a complete EHIF command set reference.

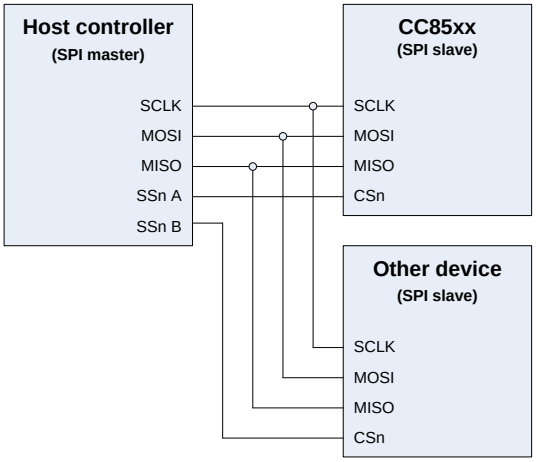


Figure 7 - SPI slave interface

2.1.3.1 Interface Description

The SPI interface consists of four signals:

Table 3 - SPI slave pin interface

CC85xx pin	Function (active level)	Description
CSn	Chip select input (active low)	When asserted, activates SPI slave. Rising edge ends or aborts an SPI operation.
SCLK	Clock Input	Serial bit clock input for SPI slave
MOSI	Slave data input	SPI Slave data input (sample on rising edge of SCLK)
MISO	Slave data output	SPI Slave data output (updated on falling edge of SCLK) Pulled up when CSn is deasserted (depending on the device configuration).

The SPI slave interface is active whenever the CSn pin is asserted. Asserting CSn will also wake up CC85xx from low-power modes and will prevent it from entering low-power modes.

When CSn is deasserted, the MISO pin is pulled up.

The interface operates on whole numbers of bytes with most significant bit first, and

- Samples serial data on MOSI on the rising edge of SCLK
- Updates serial data on MISO on the falling edge of SCLK

The interface supports SCLK frequency of up to 20 MHz, with a maximum byte rate (over any number of bytes) of 2 MHz.

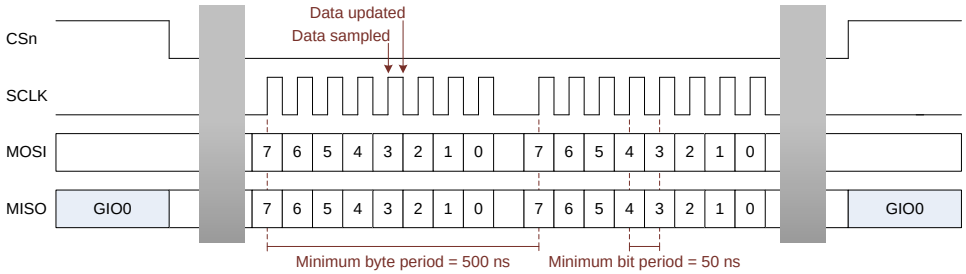


Figure 8 - SPI interface behavior

Refer to Appendix B.1 for a detailed description of SPI interface timing parameters and the set of basic SPI operations.

2.1.4 I²C Master Interface

CC85xx features a bidirectional two-wire Inter-IC (I^2C) interface, which is operated as an I^2C master and can be used to:

- Perform production programming by autonomously retrieving a firmware image from an external I^2C memory device. See appendix B.2 for further details.
- Initialize and control an external audio device, such as the TLV320AIC3101. The PurePath Wireless Configurator allows the operator to specify I^2C configuration sequences. See section 3.5.3.1 for further details.

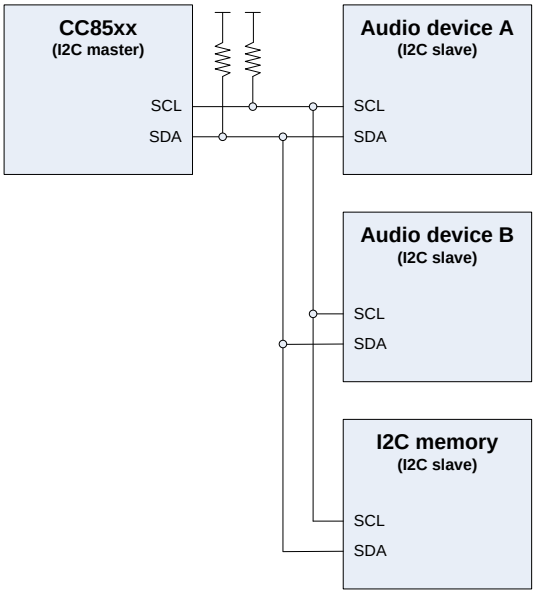


Figure 9 - I^2C bus overview

The I^2C interface supports both the 100-kbps and 400-kbps data transfer rates:

- Bootloader communication with external I^2C memory devices uses 400 kbps
- Communication with external audio devices uses the highest data rate supported by the connected device(s).

CC85xx does not support all provisions of the I²C specification:

- CC85xx can only serve as a bus master, and does not support a multi-master bus environment. A bus arbitration error will be regarded as a fatal error by CC85xx firmware, and will result in a system reset.
- I²C bus error detection (missing acknowledgment, timeout etc.) is supported; however there is no graceful error recovery. If any error is detected, the CC85xx will perform a system reset to return the system to a known state.

2.1.4.1 Interface Description

The I²C interface consists of two signals:

Table 4 - I²C master pin interface

CC85xx pin	Function (active level)	Description
SCL	Serial clock output (open-drain)	Serial clock line. Requires external pull-up resistor to be in compliance with the I²C standard.
SDA	Serial data input/output (open-drain)	Serial data line. Requires external pull-up resistor to be in compliance with the I²C standard.

The size of the pull-up resistor is application dependent. For SCL and SDA, the I²C bus specification requires rise and fall times between $20+0.1C_b$ ns (as minimum) and 300 ns (as maximum), where C_b is the total capacitance of a bus line in pF. Thus, for a bus load of 50 pF, a 2 kΩ pull-up resistor will be suitable.

Each transfer begins with a START condition and ends with a STOP condition. Data is transferred serially, bit-by-bit in units of one byte, with the most significant bit (MSB) first. After LSB of each byte is transferred, the data line changes direction to let the receiver acknowledge the transfer by transmitting an acknowledgment bit.

As bus master, CC85xx initiates all read and write transfers, and is always responsible for generating the serial clock. A bus slave can however, if needed, extend the low period of a clock cycle by holding SCL low until it is ready. As illustrated in Figure 10, SDA line must be stable while the SCL line is high (i.e. not driven), and can thus only change state while SCL is driven low.

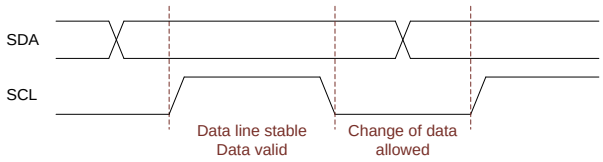


Figure 10 - I²C bit transfer

Figure 11 illustrates the transfer of a single byte, ending with acknowledge/not acknowledge by the receiver.

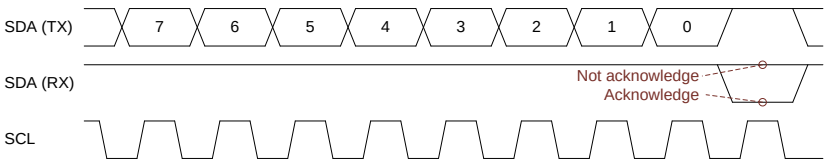


Figure 11 - I²C byte transfer

To start an I^{2}C read or write transaction, the bus master generates a START condition. It generates a STOP condition when the transaction is completed, and the bus returns to idle (i.e. SDA and SCL both high/not driven).

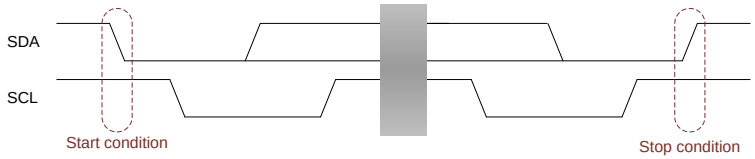


Figure 12 - I^{2}C start and stop conditions

The acknowledge bit will normally indicate successful reception of a byte. However, when the bus master is performing a read access it needs to stop the bus slave from transmitting, so that the master can generate a STOP condition. It does this by not acknowledging the last byte it wants to receive from the bus slave.

2.1.4.2 I^{2}C Operation Types

CC85xx supports I^{2}C slave devices that are identified by a 7-bit address.

Three types of I^{2}C operations are used. Note that these operations are under control by CC85xx firmware, which is generated by the PurePath Wireless Configurator.

The device configuration can affect *write-only* operations only, which are used to load register settings into external audio devices.

A **write-only** operation contains the following elements:

- CC85xx: START condition
- CC85xx: Slave address (7-bit) + write='0' (1-bit)
- CC85xx: N bytes of data
- CC85xx: STOP condition

A **read-only** operation contains the following elements:

- CC85xx: START condition
- CC85xx: Slave address (7-bit) + read='1' (1-bit)
- I^{2}C slave: N bytes of data, last byte not acknowledged by CC85xx
- CC85xx: STOP condition

A **write-read** operation contains the following elements:

- CC85xx: START condition
- CC85xx: Slave address (7-bit) + write='0' (1-bit)
- CC85xx: N bytes of data
- CC85xx: Repeated START condition
- CC85xx: Slave address (7-bit) + read='1' (1-bit)
- I^{2}C slave: N bytes of data, last byte not acknowledged by CC85xx
- CC85xx: STOP condition

2.1.5 Serial Audio Interface

CC85xx features a flexible serial audio interface that supports the I^{2}S, LJF, RJF and DSP interface formats. The interface consists of clock and data lines, and is used to transfer audio sample streams between CC85xx and external audio devices, such as CODECs, DACs, ADCs and class-D amplifiers.

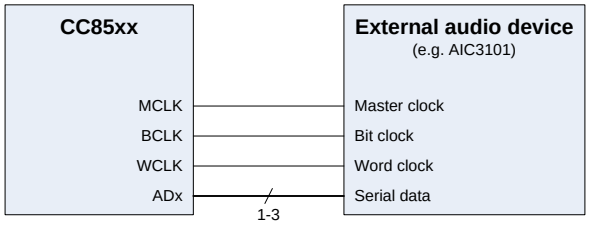


Figure 13 - Serial audio interface

2.1.5.1 Interface Description

The serial audio interface consists of 2 or 3 clock signals and 1 to 3 data signals, depending on how it is used. The clock signals can be generated either internally (by CC85xx) or externally (by the audio device or another clock source).

Table 5 - Serial audio pin interface

CC85xx pin	Function (active level)	Description
MCLK	Master clock output	Master clock for sample conversion in the external audio device. This signal is not used internally by CC85xx, and will hence never be an input to CC85xx. It is possible to select internal clock source without MCLK for external audio devices that only need BCLK.
BCLK	Bit clock input/output	Bit clock for the WCLK and the ADx signals. This signal is an input when using an external clock source, and output when using internal clock source.
WCLK	Word/sample clock input/output	Sample framing signal/clock that defines the sample frequency and the word boundaries in the serial data stream. The WCLK frequency is identical to the sample frequency. This signal is an input when using an external clock source, and output when using internal clock source.
AD0 AD1 AD2	Serial data input/output	Serial data signals responsible for transferring audio sample data. Each pin can be configured independently as input, output or unused. All pins use the same interface format (e.g. LJF).

Supported WCLK/sample rates are listed in section 2.3.2.2. The MCLK rate can be 128, 256, 384 or 512 times the WCLK rate. The BCLK rate can be 32, 64 or 128 times the WCLK rate.

CC85xx is capable of detecting noise from an external clock source on the WCLK signal. If such noise is detected it results in immediate restarting of internal audio clock generation and serial data input and output.

2.1.5.2 Supported Serial Formats

The interface supports three dual-phase formats, I^{2}S, LJF and RJF, which support 1 or 2 audio channels per ADx pin. When associating a single logical audio channel (see section 2.3.1) with an ADx pin, the sample data will be output on both I^{2}S/LJF/RJF channels (left and right).

The interface supports one single-phase format, DSP, which supports up to 8 audio channels per ADx pin. 1-6 logical audio channels (see section 2.3.1) can be associated with any of these 8 DSP audio channels.

CC85xx cannot tri-state the ADx pins, hence TDM mode is supported for ADx input pins (where only external audio devices drive these signals), but not for ADx output pins.

For all interface formats, MSB is transferred first and LSB last. Being the most flexible interface format, it is recommended to use I^2S if supported by the external audio device.

2.1.5.2.1 I²S

I^2S is a dual-phase format with a 50% WCLK duty cycle and MSB of each sample word aligned with the edge of WCLK + one BCLK period. For any given sample, LEFT channel is transferred first when WCLK is low, and RIGHT channel is transferred second when WCLK is high.

Data is sampled on the rising edge of BCLK and updated on the falling edge of BCLK.

The I^2S format is unique in the sense that CC85xx is able to auto-detect the number of BCLK periods per WCLK period and therefore supports any BCLK rate after configuration, and also variable sample resolution:

- If the sample resolution is higher than the number of bits per WCLK period, the samples are truncated
- If the sample resolution is lower than the number of bits per WCLK period, the samples are zero-padded

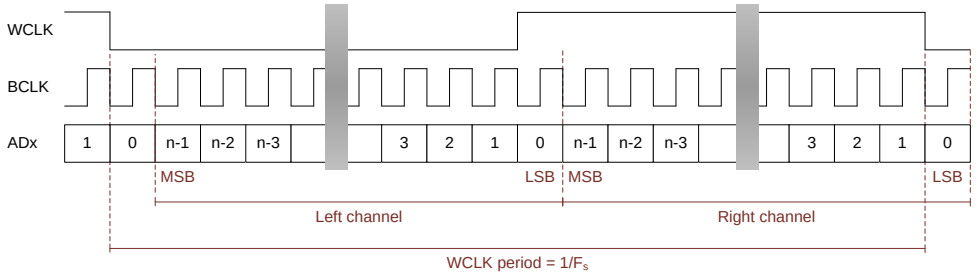


Figure 14 - I²S interface format

2.1.5.2.2 Left-Justified (LJF)

LJF is a dual-phase format with a 50% WCLK duty cycle and MSB of each sample word aligned with the edge of WCLK. For any given sample, LEFT channel is transferred first when WCLK is high, and RIGHT channel is transferred second when WCLK is low.

Configuration supports sample resolutions of 16 and 24 bits.

Data is sampled on the rising edge of BCLK and updated on the falling edge of BCLK. There is an optional idle period at the end of the clock phase.

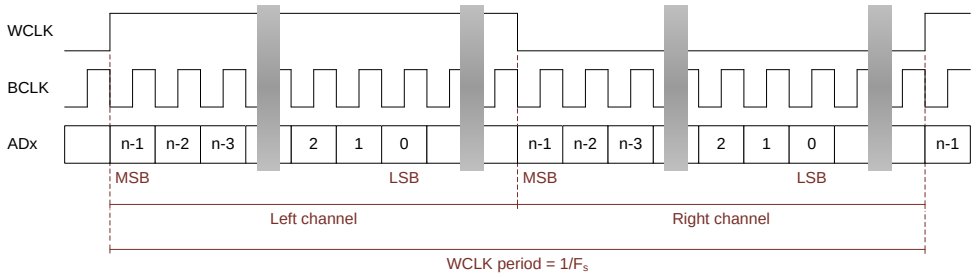


Figure 15 - Left-justified interface format

2.1.5.2.3 Right-Justified (RJF)

LJF is a dual-phase format with a 50% WCLK duty cycle and LSB of each sample word aligned with the edge of WCLK. For any given sample, LEFT channel is transferred first when WCLK is high, and RIGHT channel is transferred second when WCLK is low.

Data is sampled on the rising edge of BCLK and updated on the falling edge of BCLK.

Configuration supports sample resolutions of 16 and 24 bits. There is an optional idle period at the start of the clock phase.

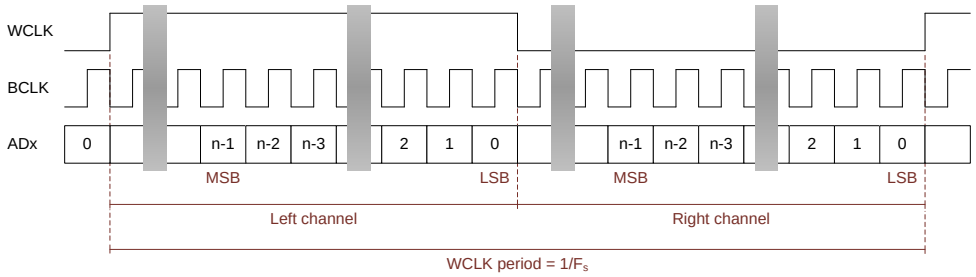


Figure 16 - RJF interface format

2.1.5.2.4 DSP

DSP is a single-phase format where WCLK is high for one BCLK period, followed by each audio channel back-to-back, followed by an idle period until the next WCLK period begins.

Data is sampled on the falling edge of BCLK and updated on the rising edge of BCLK.

Configuration supports sample resolutions of 16 and 24 bits. There is an optional idle period at the end of the clock phase.

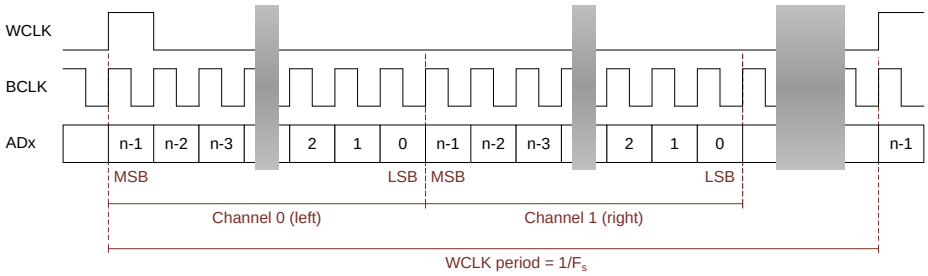


Figure 17 - DSP interface format (showing 2 first of 8 possible channels)

2.1.6 CC259x Range Extender Control Interface

The range extender control interface provides a seamless interface to the CC259x device family. Using a range extender together with the CC85xx will significantly increase the sensitivity and output power, and thus improve RF link budget and range.

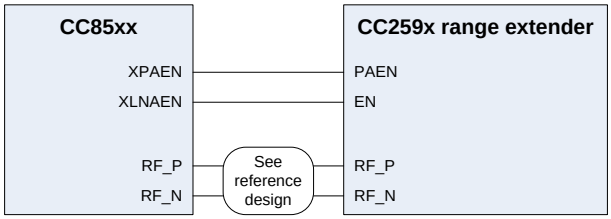


Figure 18 – CC259x Range extender control interface

2.1.6.1 Interface Description

The range extender interface consists of two control signals:

Table 6 - Range extender control pin interface

CC85xx pin	Function (active level)	Description
XPAEN	PA enable output (active high)	Enables the CC259x PA for packet transmission.
XLNAEN	LNA enable output (active high)	Enables the CC259x LNA for packet reception.

The XPAEN signal goes high slightly before preamble transmission begins and remains high slightly after the last bit of the packet has been transmitted.

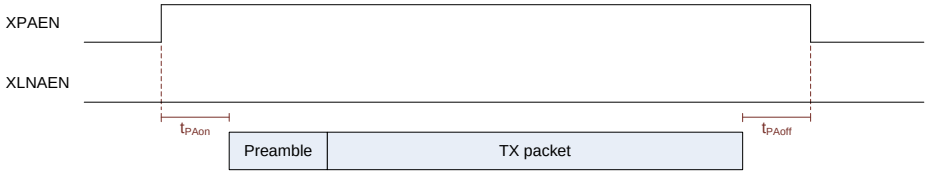


Figure 19 - XPAEN timing

The XLNAEN signal goes high slightly before earliest preamble reception is expected and remains high until slightly after the last bit of the packet has been received.

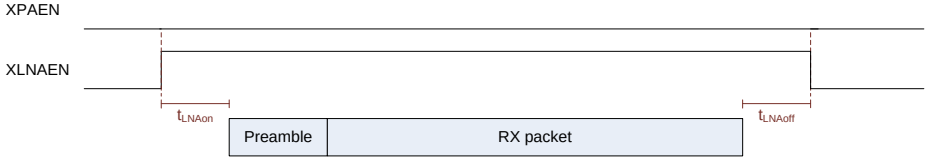


Figure 20 - XLNAEN timing

When the range extender control interface is not in use, the pins can be left unconnected or be used for other purposes (as GIO14 and GIO15).

2.1.7 Battery Voltage Monitoring Interface

CC85xx features a voltage sensing pin that can be used for battery supervision. Thresholds for starting up and shutting down can be configured in the PurePath Wireless Configurator. This is typically used to:

- Prevent Li-Ion and similar battery types from discharging below the level where it causes damage to the cells
- Shut down the system before supply under voltage reduces CC85xx performance or causes the external audio device (or other external circuitry) to misbehave

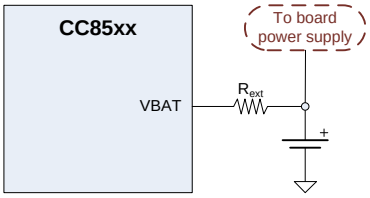


Figure 21 - Battery voltage monitoring

The shut-down occurs well in time before the battery voltage drops below a critical level, and includes the normal shut-down procedures for the external audio device and other external circuitry.

It is also possible to enable an LED-based warning when the battery voltage drops below a configurable threshold.

2.1.7.1 Interface Description

The CC85xx VBAT pin is connected to the battery through an external series resistor, R_{ext}. The resistor is needed to divide down the battery voltage to a level that can be measured by the internal circuitry. The value of the external resistor is application specific, and must be specified at configuration time.

The measurement is done as 6-step binary search, which produces one of 64 values. There are significant chip variations with respect to measureable voltage range. The PurePath Wireless Configurator calculates the possible range variations when the R_{ext} value is specified. The effect of R_{ext} tolerance can be found by entering its minimum and maximum values.

The VBAT pin can tolerate voltages up to 4.5 V. If the battery voltage can be higher than this (including while charging), a 3.9 V zener diode to GND should be inserted between the VBAT pin and the external series resistor, R_{ext}. Note that doing this will affect the measured voltage, and must be compensated for when specifying R_{ext} value and voltage thresholds at configuration time and when reading the voltage through the external host interface.

If power is controlled using a switch in series with the battery, this switch should be inserted between the battery and R_{ext} so that current does not leak while powered down.

The VBAT pin should be left unconnected or tied to ground when not used.

2.1.8 USB Interface

CC85x1 features a 12 Mbps full-speed USB function with associated internal hardware peripherals for audio clock synthesis and a FIFO controller to handle isochronous endpoints.

2.1.8.1 Interface Description

The USB interface consists of two signals:

Table 7 - USB function pin interface

CC85xx pin	Function (active level)	Description
USBP	Data line D+	Differential data line, D+. Pulled high through a 1.5 kΩ resistor to signalize that the CC85xx is connected to the USB bus.
USBN	Data line D-	Differential data line, D-.
GIO9	Controls pull-up on data line D+	Ensures that the CC85x1 does not signalize its presence on the USB bus until it is ready to enumerate. Also provides device disconnection during programming to ease product evaluation and customization.

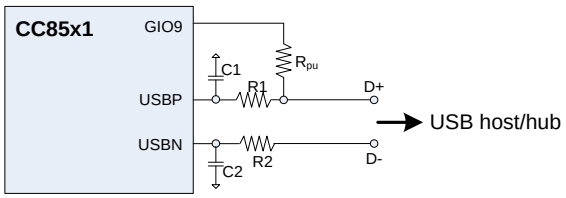


Figure 22 – USB interface

Refer to the USB specification and/or reference designs for external component values.

When not in use, USBN may be tied to GND and USBP may be tied to IOVDD. There is an internal weak pull-down resistor to GND on USBN and an internal weak pull-up resistor to IOVDD on USBP.

2.1.8.2 Endpoint Configuration

CC85x1 hardware implements the following USB endpoints:

- EP0: Control endpoint (32-byte single-buffered FIFO)
- EP1: Isochronous OUT endpoint for audio streaming to protocol slave (double-buffered FIFO)
- EP2: Isochronous IN endpoint for audio streaming from protocol slave (double-buffered FIFO)
- EP3: Bulk/interrupt IN endpoint for play control (+ keyboard) or custom-defined HID reports (32-byte single-buffered FIFO)
- EP4: Bulk/interrupt IN endpoint for mouse or custom-defined HID reports (32-byte single-buffered FIFO)
- EP5: Bulk/interrupt OUT endpoint, currently unused (32-byte single-buffered FIFO)

2.1.9 Antenna Switch Control Interface

CC85xx can control an antenna switch to support antenna diversity with two antennas. This can significantly reduce RF multi-path fading effects and improve the performance of the adaptive frequency hopping mechanism.

Only **protocol slaves** support the antenna switch control.

The antenna selection algorithm works best with protocol masters running FW 1.3 or later, as it requires extended master packet preamble to select antenna on per-packet basis. The extended preamble is enabled automatically when an antenna diversity enabled protocol slave joins the network.

2.1.9.1 Interface Description

The antenna switch interface consists of two signals:

Table 8 – Antenna switch function pin interface

CC85xx pin	Function (active level)	Description
XANTP	Antenna select (positive level)	Select antenna A when XANTP is low, select antenna B when XANTP is high.
XANTN	Antenna select (negative level)	Select antenna A when XANTN is high, select antenna B when XANTN is low.

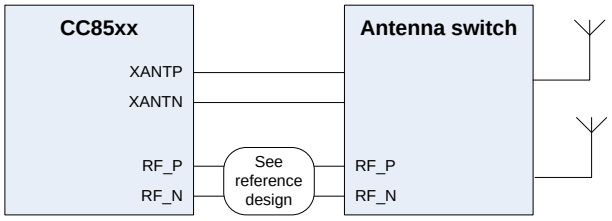


Figure 23 - Antenna switch control

When not in use, XANTN may be tied to GND and XANTP may be tied to IOVDD. There is an internal weak pull-down resistor to GND on XANTN, and an internal weak pull-up resistor to IOVDD on XANTP.

2.2 Wireless Network Overview

This section provides a brief introduction to PurePath Wireless networks and their main features. Refer to section 2.4 for further details on the radio protocol, and section 3.4 for details on practical network management.

2.2.1 Network Topology

PurePath Wireless networks use a star topology, consisting of a single protocol master (PM) and one or more protocol slaves (PS). All communication in the network is either to or from the protocol master; there is no direct communication between protocol slaves.

Figure 24 illustrates how information/data flows in the network:

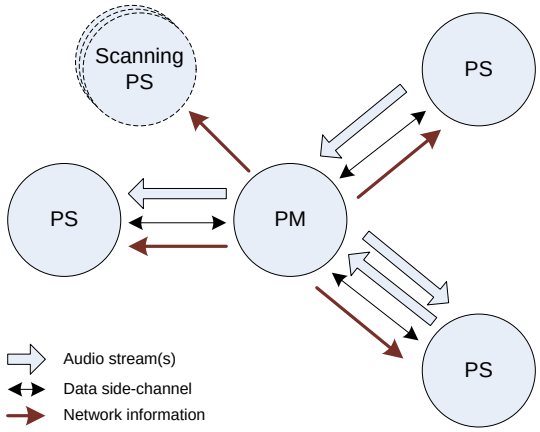


Figure 24 - Network topology

The maximum number of protocol slaves supported by a network is set on the protocol master at configuration time. This value determines several fixed parameters for the network and therefore affects audio streaming robustness and data side-channel throughput.

The network supports up to four audio streams, which may go to and from the protocol master, with no inherent restrictions. However, as for maximum slave count, the number of audio channels streamed and used streaming formats do have a significant impact on audio streaming robustness.

The data side-channel can be used between a host-controlled protocol master and host-controlled protocol slaves, or be disabled at configuration time if not needed.

2.2.1.1 Device Identification

A CC85xx device is identified by three 32-bit values:

- A globally unique **device ID**, which is set in hardware production by Texas Instruments. This value is used to identify each device in a network, and the protocol master's device address is also referred to as the **network ID**.
- A unique **manufacturer ID**, based on the device ID of a purchased device, which identifies the manufacturer of the product. This value is used for identification during network scanning, and can also be used for network pairing filtering (i.e. reject other manufacturers).
- A manufacturer-specified **product ID**. This value is used for identification during network scanning, and can also be used for network pairing filtering (i.e. only accept certain products from the given manufacturer).

Refer to section 3.4.1 for further details.

2.2.1.2 Network Information

The protocol master broadcasts information about its network to currently joined protocol slaves, and also to protocol slaves that are searching for a network to join. This information includes, but is not limited to:

- The protocol master's device address, manufacturer ID and product ID
- Device addresses for each protocol slave that is currently part of the network
- Supported logical audio channels (see section 2.3.1 for further details) and the used streaming format
- Audio sample rate and latency
- Volume control information
- Network power control

2.2.1.3 Audio Stream

A CC85xx device can be an audio **producer** (i.e. its audio interface is connected to the source of one or more audio channels), an audio **consumer** (i.e. its audio interface is connected to the destination of one or more audio channels), or both.

Each audio channel supported by the network is treated as a separate data stream over the air and is subject to individual flow control, acknowledgement and retransmission. Audio channels produced at the protocol master can be consumed by multiple protocol slaves (using multicasting where each consumer individually acknowledges) whereas an audio channel produced by a protocol slave is consumed only by the protocol master.

Audio streaming is described in further detail in section 2.3.

2.2.1.4 Remote Control

Protocol slaves can transmit remote control information to the protocol master.

When using the pre-defined remote control scheme, the information contains codes for (up to 7 in total) currently pressed remote control command keys (volume up/down, play/pause etc.) and/or standard keyboard keys. The information can optionally also contain mouse movement (two axis) and button states. See sections 3.3.3 and 3.6.3 for details.

For protocol masters supporting multiple protocol slaves, the key/button states from each contributor are OR'ed together. Mouse movements received from multiple protocol slaves are summarized.

Mouse movements are transferred using an unacknowledged protocol that prevents loss of movement.

The key state information is transmitted continuously by the protocol slaves. If remote control information is not received from a protocol slave (e.g. it has been switched off, or it has lost network connection), all remote control states for that protocol slave will become inactive after a 100 ms timeout.

Alternatively, the remote control information can carry proprietary data, to be used for instance to generate custom-defined HID reports on a USB-based protocol master. See sections 2.5.3, 3.3.3 and 3.6.3 for details.

2.2.1.5 Data Side-Channel

Each host-controlled protocol master-slave pair can establish an ordered-sequence, assured delivery datagram connection in each direction. This so-called data side-channel can be used by host processors during audio streaming for various application-specific control or information purposes.

Operation of the data side-channel is described in further detail in section 3.4.4.

2.3 Audio Streaming

This section describes how audio streaming is achieved.

2.3.1 Audio Stream Routing

The purpose of audio stream routing is to deliver each audio channel on the producer to its correct destination on the consumer(s), and ensure that this works well independently of the protocol master and slaves participating in the network, for random slave types (e.g. a headphone or two mono speakers) and also across different manufacturers.

The routing is accomplished by assigning each audio channel a logical channel ID, so that the producer and the consumer have a common understanding of which audio stream is which when transmitted across the PurePath Wireless network. Each network node maps these logical channels to the I^2S/LJF/RJF or DSP channels on its local serial audio interface or the USB audio device channels.

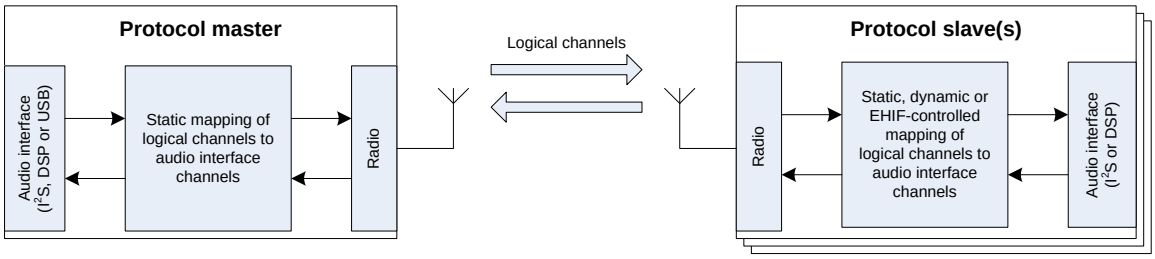


Figure 25 - Logical channel mapping

The mapping between logical channels and audio interface channels is statically defined at configuration time for protocol masters. For protocol slaves, the mapping is more flexible:

- Autonomous
  - The configuration can be static, as for protocol masters
  - Mono channel and certain stereo slave application roles: Dynamic selection via pin-control, where 1 to 3 GIO pins are used to select from a set of 2, 4 or 8 channels (defined at configuration time), or where a button is used to cycle through 2 to 8 pre-defined channel configurations
  - Mono channel microphone application roles: Automatic dynamic channel selection, based on which channels are already in use
- Host-controlled: The host processor assigns logical channels to a set of predefined audio interface channels. This allows for static, dynamic or protocol master-dependent setup of channel mapping.

Table 9 - Logical channels

Protocol master to protocol slave		Protocol slave to protocol master	
ID number	**Logical channel**	**ID number**	**Logical channel**
0	Front/primary left	10	Input left
1	Front/primary right	11	Input right
2	Rear/secondary left	12	Microphone 0 (left)
3	Rear/secondary right	13	Microphone 1 (right)
4	Front center	14	Microphone 2
5	Subwoofer	15	Microphone 3
6	Side left		
7	Side right		
8	User defined 0		
9	User defined 1		

The PurePath Wireless network will only transfer logical channels that exist on both the producer and the consumer. If a logical channel only exists on the producer, it will not be transferred and all samples will be discarded. If a logical channel only exists on the consumer, the consumer will output silence on the audio device channel it is mapped to.

The set of logical channels must be selected carefully, and it is recommended to follow these guidelines for best compatibility (with both own and other manufacturers' products):

- Make sure that the network roles (i.e. protocol master or protocol slave) for the devices are correctly selected. Protocol masters and protocol slaves have different capabilities and characteristics (for instance, a CC85x1 device implementing a USB soundcard must always be the protocol master).
- For mono speakers to be used in a home entertainment system, it is recommended to use dynamic audio channel selection. This can be operated by the end-user through a DIP-switch or similar.
- For stereo speakers, including headphones and headsets, use the front/primary left and front/primary right channels
- For base stations and speakers/headphones supporting two independent stereo streams, use front/primary left/right for the first stereo stream and rear/secondary left/right for the second stereo stream
- A subwoofer shall always use the subwoofer channel.
- A microphone shall always use a microphone channel. It is recommended to use dynamic channel selection for standalone mono microphones, and microphone 0 for headsets.
- The user defined channels should only be used when necessary, and in combination with manufacturer (and product) filtering.

The device type of the CC85xx determines the maximum number of channels that can be transferred. See Table 1 for device type capabilities.

If two protocol slaves produce the same logical channel, ownership is given on a first-come first-served basis. The "next" protocol slave in line will get ownership of the logical channel when the first protocol slave leaves the network, or stops streaming of the audio channel.

2.3.2 Network Audio Clock

In a wired digital audio system the reference clock that determines the audio sample rate is normally distributed in parallel with or embedded into the signal from the audio source. Similarly, when transferring audio samples wirelessly, there is need for a mechanism to distribute the reference clock to maintain a synchronous system.

The PurePath Wireless platform uses a mechanism called wireless clock distribution, which ensures that:

- All protocol slaves use the same sample rate as the protocol master. The platform supports sample rates from a predefined set of discrete values (see Table 10).
- All audio streams are delayed by the same constant amount, as defined at configuration time. This delay is referred to as the audio latency, and there is one delay value (in number of samples) for each sample rate.

2.3.2.1 Wireless Clock Distribution

The protocol master's audio clock, which can be generated internally by CC85xx or by an external clock source, is used as reference for all nodes in the network. The wireless distribution of this reference clock is achieved through a phase and frequency locked loop implemented in each protocol slave that uses clock phase values communicated by the protocol master as a reference clock and clock phase values from its local, internally generated, clock as feedback.

The nominal sample rate and audio latency is broadcasted by the protocol master as part of the network information (ref. section 2.2.1.2). These values are used by the protocol slaves to initialize their internal clock generators.

Each node in a network runs a counter, which is connected to the local clock source. The actual goal of the clock distribution mechanism is to keep all these counters synchronized. When the protocol master transmits the packet synchronization word, it captures the local counter value, and includes it later in the same packet. Similarly, when a protocol slave receives the synchronization word, it captures the value of its own counter.

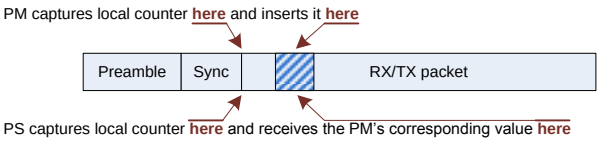


Figure 26 – Counter capturing

After reception of the protocol master's counter value, the protocol slave calculates the difference between the counter values. This value, which is referred to as the phase error, is then used as input to a soft-PLL, which in turn controls the rate of the locally generated clock. The locally generated clock is adjusted so that the phase error approaches zero.

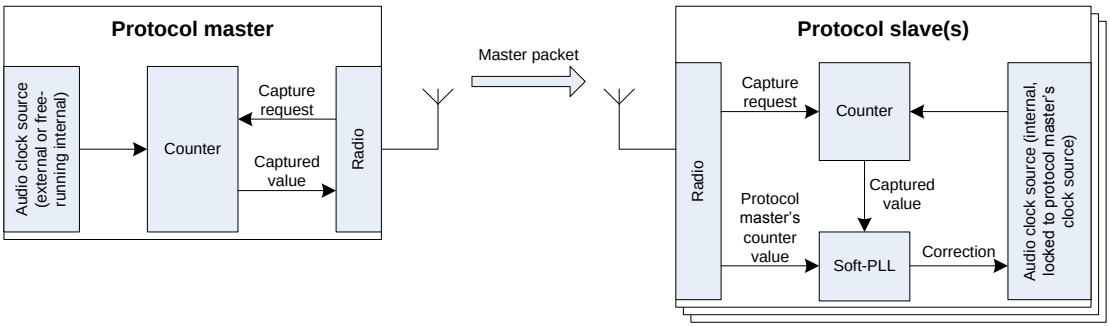


Figure 27 - Clock distribution overview for non-USB system

The counter values have sufficient precision that low-jitter replication of the reference clock is possible. The resulting clock distribution ensures that the latency variation across the wireless audio network, as seen on the pins on the audio serial interfaces, will be less than ±1 sample for an audio input clock with low to moderate jitter.

USB devices implement sample clock synthesis, controlled by USB start-of-frame timing and the number of samples received per USB frame from the USB host. To ensure constant audio latency, the mechanism includes error handling for missing start-of-frame and other packet errors.

2.3.2.2 Supported Sample Rates

The PurePath Wireless platform supports a set of discrete, predefined sample rates. The list of supported sample rates for a particular device may be reduced by limitations of the selected external audio device. For instance, some audio devices only support sample rates that are multiples of 8 kHz and 11.025 kHz.

Table 10 - Supported sample rates

Nominal sample rate	Supported by USB devices
32.000 kHz	Yes
40.275 kHz	No
44.100 kHz	Yes
48.000 kHz	Yes

When using an external clock source, the protocol master will detect the supported sample rates and automatically switch sample rate for the entire network. The frequency of the external clock must be within a window of ±2000 ppm around the desired nominal sample rate. The frequency of the external clock can drift within this window at a maximum rate of ±100 ppm per second.

When the protocol master uses its internal clock generator, only one of the above sample rates is supported and must be set at configuration time. The tight tolerances of the crystal frequency required for correct radio operation ensures that the generated reference sample rate is within ±50 ppm of the nominal sample rate.

Further details on sample rate control are provided in section 3.5.2.

2.3.2.3 Audio Latency

The PurePath Wireless platform delays the audio stream to be able to retransmit parts of the stream that are lost due to RF interference and RF fading effects. Hence, selecting the audio latency is a trade-off between the audio stream delay and the robustness of the link; there is room for more retransmissions with a higher audio latency setting.

The audio latency is defined as the delay introduced by CC85xx when replacing a wired serial audio interface (I^2S, LJF, RJF or DSP) connection between an audio producer and an audio consumer. Hence, latency values specified in the PurePath Wireless Configurator do not include delays introduced by the external audio device (e.g. a DAC) connected to CC85xx.

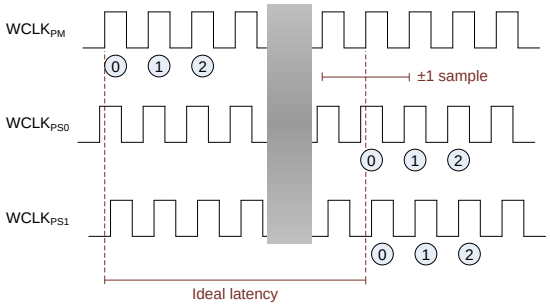


Figure 28 - Audio latency

The latency is the same for all audio streams in the wireless audio network regardless of direction. The audio latency used in an audio network is announced by the protocol master, and can be individually set for each supported sample rate with a granularity of 64 samples in the range 512 to 2048 samples. The audio latency for each sample rate is a static configuration on the protocol master.

Note that it is always recommended to use the highest possible latency setting for a given application. The lowest audio latency settings are restricted as shown in Table 11:

Table 11 – Minimum latency vs. possible configurations

Using PCM16/PCME24 streaming formats	Number of audio channels	Minimum latency
No	1	512
No	2 or more	640
Yes	1	640
Yes	2 or more	768

The wireless clock distribution mechanism described earlier ensures that the latency variation between different nodes in a wireless audio network is kept within ±1 sample for an audio input clock with low to moderate jitter.

The filter processing required in the PCMLF format in order to do interpolation and decimation adds another 37 samples to the audio latency.

2.3.3 Streaming Formats

Each raw sample stream undergoes processing before and after transmission over the wireless link:

- To be able to support error detection and retransmission of missing or corrupted pieces of the sample stream, it is divided into small blocks of samples called slices. Each audio slice has a sequence-identifying index number and a checksum for integrity checking.
- Each audio slice is encoded in a streaming format specified in the protocol master's device configuration. This provides a tradeoff between streaming capabilities, audio quality and streaming robustness. The streaming format is specified individually for each audio channel.

The following streaming formats are available:

Table 12 - Streaming format overview

Format	Description	Extra Delay	Data rate @ 48 kHz
PCM16	Standard 16-bit lossless 2's complement PCM samples	None	768 kbps
PCMLF	PCM16 decimated-by-four at producer and interpolated-by-four at consumer (for low frequency audio channels)	37 samples	192 kbps
SLAC	16-bit or 24-bit compressed full-fidelity audio stream using proprietary SLAC codec	None	232 kbps (average)
PCME24	24-bit companded PCM samples, offering 15-bit signal-to-noise ratio and 24-bit dynamic range	None	768 kbps

Note the following guidelines/limitations when selecting streaming formats:

- 1 or 2 channels: No restrictions
- 3 channels: At most 2 channels should use PCM16/PCME24 to achieve good robustness
- 4 channels: Only SLAC and PCMLF may be used
- Bi-directional audio streaming applications: Only PCM16 and SLAC may be used

2.3.3.1 PCM16

The PCM16 format transfers each sample without any modifications, with a resolution of 16 bits.

The format is interleaved by a factor 2 to reduce the risk of drop-outs under poor wireless link conditions. If the throughput over time is less than required (e.g. less than 768 kbps per PCM16 channel at 48 kHz), the producer prioritizes transmission of even audio samples, and the consumer approximates any missing odd samples by linear interpolation (average of the two neighbor even samples). This mechanism provides better quality-of-service at the fringe of the RF coverage area, but is normally not active.

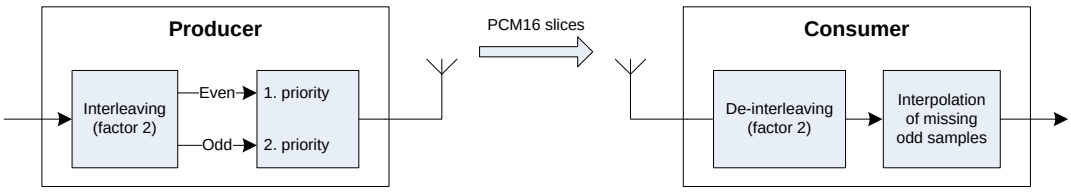


Figure 29 - PCM16 slice processing

2.3.3.2 PCMLF

The PCMLF format is designed specifically for subwoofer applications. The input signal is decimated down by a factor 4 in the audio producer and interpolated up by a factor 4 in the audio consumer. The processing includes anti-aliasing filters before decimation and after interpolation to allow input of a full spectrum audio signal without degradation of the low frequency content due to spectral aliasing or replication.

The reduced data bandwidth of this format provides more time for retransmission and thus improves RF coverage.

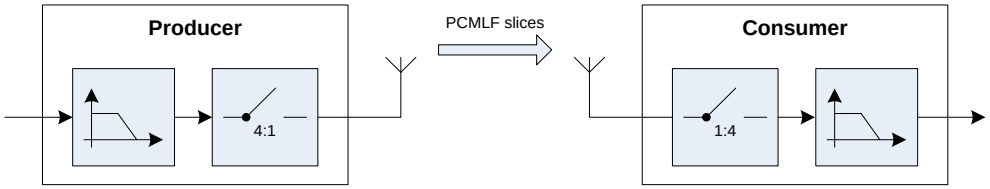


Figure 30 - PCMLF slice processing

Refer to appendix C for detailed filter characteristics.

2.3.3.3 SLAC

The SLAC format is intended for audio streaming where audio quality can be traded against increased link robustness. The SLAC algorithm (Slightly Lossy Audio Codec) is a proprietary audio compression algorithm, based on compression techniques widely used in lossless audio codecs (e.g. FLAC or Monkey) together with quantization. It is designed to give near-transparent audio quality at a considerably lower data rate.

- Each channel is sub-divided into blocks with individual and independent encoding/decoding (i.e. no intra-channel or intra-block dependency). This allows for easy re-transmission and no “domino error effects” in the audio stream.
- The algorithm uses a variable-length entropy code that allows moments of high audio activity to expand in size beyond the target bit-rate, in order to ensure a good listening experience
- For a single audio slice the average number of bits per sample cannot exceed 8, whereas on average the bit rate is 4.8 bits/sample @ 48 kHz and 5.3 bits/sample @ 44.1 kHz.
- Noise shaping is employed to push quantization noise to psycho-acoustically less audible frequencies.
- The algorithm can encode 24-bit audio input by applying a similar technique as the PCME24 format (described below) before SLAC encoding and after SLAC decoding. This behavior is activated automatically by configuring the audio interface for 24-bit resolution.
- The SLAC algorithm will, as most compression algorithms, give a frequency response that is not flat. Therefore, SLAC evaluation should not be done by frequency sweeping, but rather by listening to real audio data.

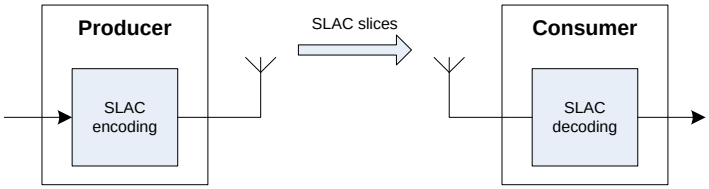


Figure 31 - SLAC slice processing

2.3.3.3.1 SLAC Encoding

The SLAC encoder works on blocks of 64 samples, and provides output blocks of up to 64 bytes. It uses signal prediction, quantization and variable length entropy codes to reach an average bit-rate of 232 kbit/s pr. channel. If there are abrupt transients in the input signal (e.g. movie explosions) the encoder will occasionally encode to an output block that is larger than the 64 byte maximum, and an additional encoding pass will be performed with more conservative parameters to ensure that the output block size requirement is met.

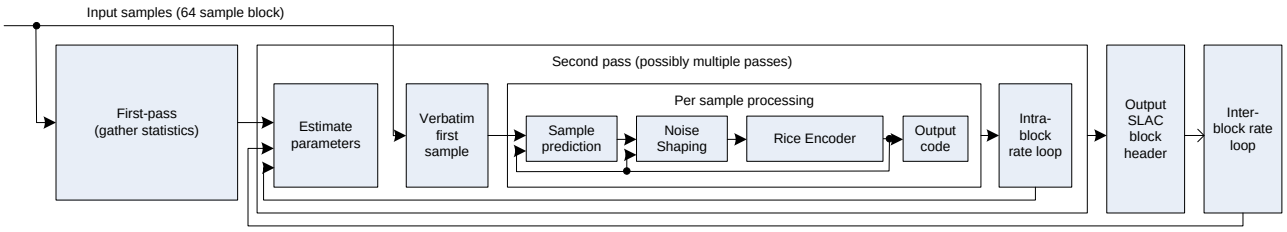


Figure 32 – SLAC encoding (16-bit)

2.3.3.3.2 SLAC Decoding

The SLAC decoder is less complex than the encoder, and it uses only one pass to decode a SLAC block. No data processing is performed other than the decoding itself. All parameters needed for decoding a slice of SLAC audio data is embedded into the slice itself.

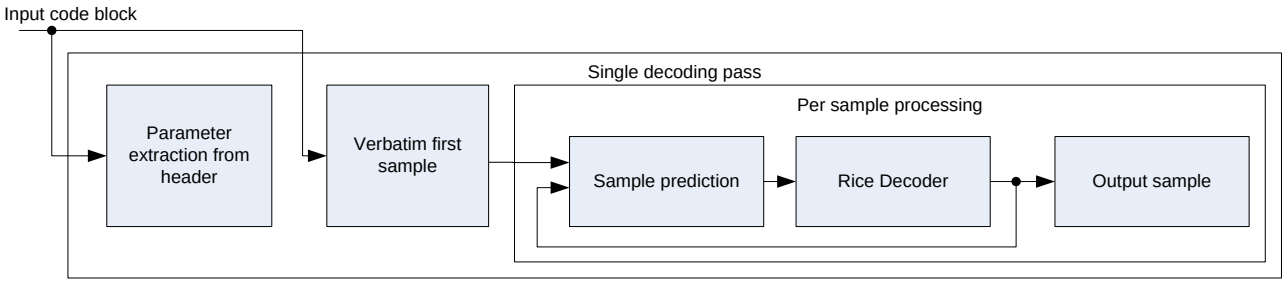


Figure 33 – SLAC decoding (16-bit)

2.3.3.4 PCME24

The PCME24 format interleaves the incoming 24-bit samples, as done by the PCM16 format, and then converts four and four 24-bit samples into micro-blocks of four 15-bit sample values and a common 4-bit shift value, thereby using the same data bandwidth as PCM16. By shifting the 15-bit window within the 24-bit sample space according to the highest sample value in each 4-sample block, the format achieves 15-bit signal-to-noise ratio and 24-bit dynamic range.

With identical data bandwidth and transport mechanism, the PCME24 format offers link robustness identical to that of the PCM16 streaming format.

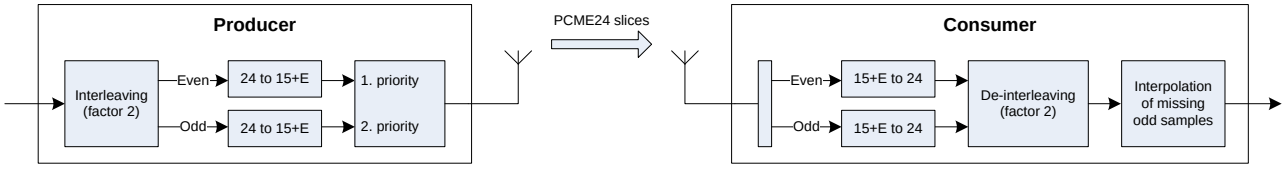


Figure 34 – PCME24 slice processing

2.3.3.5 Handling of Drop-Outs

If the wireless link fails to transfer any even PCM16/PCME24 slices, any PCMLF slices or any SLAC slices, all output audio channels on the failing receiver node will be muted. The channels stay muted until the quality of service again is found acceptable (i.e. there have been no missing audio slices for a certain period of time, typically 2-300ms depending on sample rate).

The muting and un-muting operations are done in a soft manner over valid audio samples in order to avoid abrupt transitions. It is, however, possible to disable the soft fade-out for improved audio streaming robustness corresponding to a 64 samples audio latency increase.

2.4 Radio Protocol

The PurePath Wireless platform uses a proprietary radio protocol designed specifically for audio streaming. The protocol supports a star network topology with a single protocol master (PM) node and up to four protocol slave (PS) nodes. All traffic is limited to burst transfers within constant duration timeslots, where the intra-timeslot timing is controlled by the protocol master and protocol slave nodes are only allowed to transmit data at times explicitly stated by the protocol master. All traffic goes from or to the protocol master, never between protocol slave nodes.

2.4.1 Network Management

Each CC85xx device is identified by a unique 32-bit device address, which is stored in the hardware by Texas Instruments during chip production. The protocol master's device address is used as network ID, and is stored in the protocol slaves during power-down so that the network can be re-established when operation is resumed.

A network is normally established through button-based pairing (i.e. pressing a physical button on the protocol master and another physical button on the protocol slave within a configurable interval) and/or proximity-based pairing. It is also possible to program the network ID into the protocol slaves during production, or implement a custom solution for network establishment with host-controlled operation.

Each device is identified by a manufacturer ID and a product ID. This allows for filtering during pairing, for instance so that a headphone product only allows pairing with a specific base station made by a specific manufacturer.

The protocol master broadcasts network information (see section 2.2.1.2) so that protocol slaves can search passively for suitable networks. This search operation will therefore not cause interference with existing networks.

A protocol slave joins a protocol master's network by transmitting a so-called Slave Join Packet. A random-backoff mechanism resolves situations where multiple protocol slaves simultaneously attempt to join the same network. A protocol slave leaves a network by ceasing all communication with the protocol master, and is then ejected from the network after a certain timeout. Once ejected (the protocol slave is no longer listed in the broadcasted network information) a new Slave Join Packet is required to re-join the network.

The practical aspects of network management for autonomous and host-controlled operation are described in detail in section 3.4.

2.4.2 Anatomy of a Timeslot

PurePath Wireless divides time into so-called timeslots, during which all nodes in the audio network operate on the same RF channel and each node transmits one packet. The duration of a timeslot is nominally 2.5 ms, but this parameter depends on the protocol master's configuration.

To achieve robustness against interferers and frequency-selective fading, each successive timeslot is transmitted at a different RF channel following a deterministic sequence. See section 2.4.3 for further details.

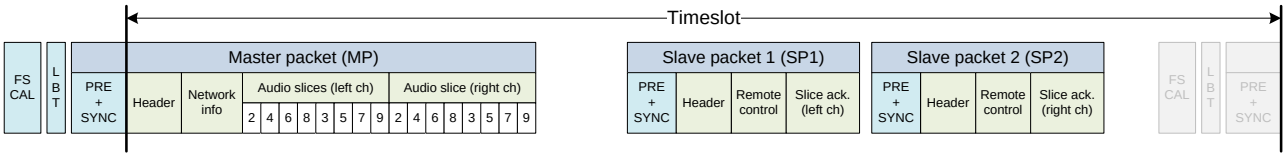


Figure 35 - Anatomy of a timeslot with two protocol slaves, network full (not drawn to scale)

The start of a timeslot is defined as the point in time when the protocol master has just transmitted its packet synchronization sequence. The protocol master dictates the timing within each timeslot and informs the protocol slaves if and when they may transmit and when the next timeslot will start. The sequence of events during a timeslot is illustrated in Figure 35 and summarized below for a three-node audio network:

- **Frequency synthesizer calibration** is performed just prior to the start of the timeslot
- The protocol master performs an **energy measurement** in the RF channel to see whether it is already in use
- If the energy is below a threshold, the **protocol master transmits its packet**. This marks the start of the timeslot. The protocol master packet contains:
  - Information about the audio network
  - Data side-channel datagram to a single protocol slave, if any
  - Audio slice acknowledgments for each audio channel the protocol master consumes
  - Audio slices for each active audio channel that the protocol master produces
- Following the protocol master packet is the **slave join packet slot**. This is reserved for protocol slaves that wish to join the audio network. If the audio network has the maximum number of supported protocol slaves, this slot is removed.
- Then each of the **protocol slaves transmits their packets** at the time offset in the timeslot announced by the protocol master in its packet. Each protocol slave packet contains:
  - Remote control information
  - Data side-channel datagram to the protocol master, if any
  - Audio slice acknowledgments for each audio channel the protocol slave consumes
  - Audio slices for each active audio channel that the protocol slave produces.

2.4.3 Adaptive Frequency Hopping

The purpose of using frequency hopping in a radio system is to provide diversity that allows data throughput to be maintained even if interfering radio systems or the physical environment (e.g. multipath fading) render some RF channels unusable. In the 2.4 GHz ISM band, the sheer amount of radio systems and the severity and dynamic nature of indoor fading phenomena in typical operating environments require the use of this kind of diversity if a minimum data throughput is to be assured (as audio streaming requires).

Frequency hopping systems can either implement a fixed sequence of channel hops or adapt its hopping sequence dynamically to the changing environment it operates in. In order to maximize its own chances of delivering audio data in time and to co-exist amicably with other fixed-frequency or adaptive frequency-hopping systems, PurePath Wireless uses an adaptive frequency hopping (AFH) scheme that adapts to changing conditions within tens of milliseconds.

PurePath Wireless divides the 2.4 GHz band into 18 RF channels with 4 MHz bandwidth. The protocol master controls the adaptive frequency hopping scheme for the audio network, and maintains a table with an entry for each RF channel and an associated quality-of-service (QoS) estimate for each. Each time an RF channel is used the QoS estimate is updated based on what happens during the timeslot.

The frequency hopping algorithm separates the 18 RF channels into two sets:

- A set of 4 *active channels*
- A set of 14 *trial channels*

The active channel set contains the preferred RF channels that have proven that they provide sufficiently good quality-of-service. The trial channel set contains the remaining RF channels that are only evaluated occasionally in order to be able to maintain an accurate picture of their quality-of-service. If the QoS estimate of an RF channel in the active set goes beyond a minimum threshold this channel is swapped out with the RF channel in the trial channel set that has the best QoS estimate. Other factors play in when selecting a new RF channel to the active channel set, such as trying to maintain a certain minimum distance in frequency between the different active channels.

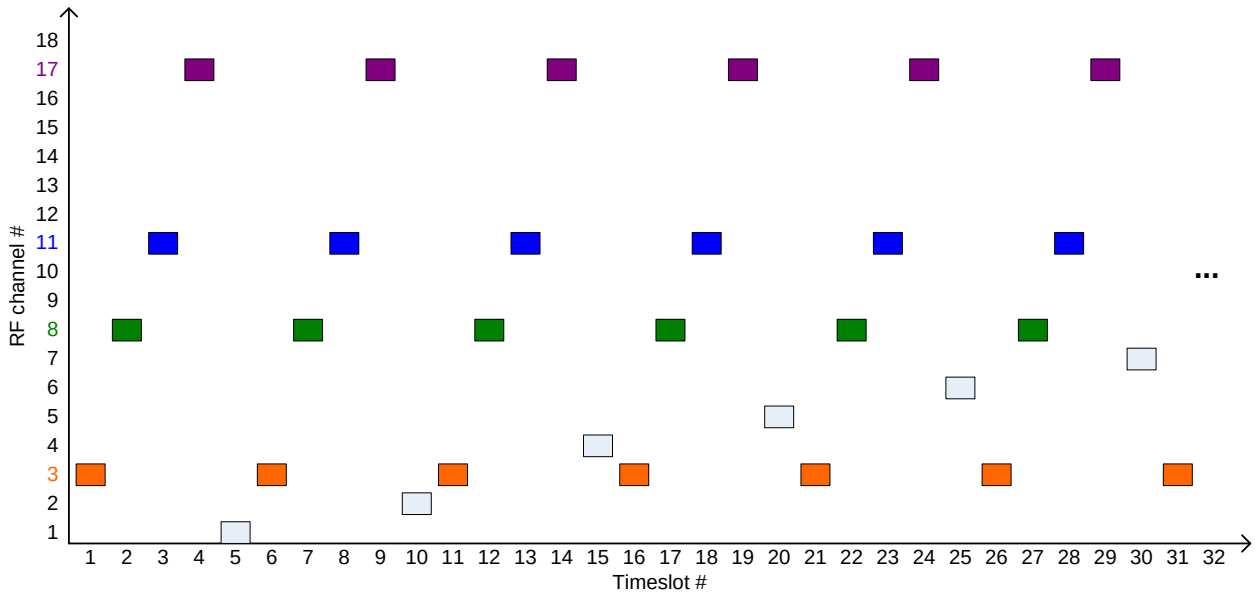


Figure 36 – Example of AFH hop sequence (active set in color, trial set in black/gray)

The frequency hopping algorithm, when using all 18 channels and no swaps between the active and trial channel sets occur, goes through a sequence of 70 hops over the course of which every RF channel has been used.

- This 70-hop *macrosequence* consists of 14 repetitions of a
  - 5-hop *microsequence* during which
    - Each of the four active RF channels are used once
    - One of the trial RF channels is used once
      (cycling through all trial channels over the course of a macrosequence)

Figure 36 illustrates this concept. This gives an average steady-state RF channel usage of:

- Each of the four active channels are used 20% of the time
- Each trial channel is used 1.43% of the time

2.4.3.1 Manual Frequency Planning

If a known static high duty-cycle interferer is present in the end-product, manual frequency planning can be used to avoid this interferer. This is done by excluding RF channels to be used, effectively reducing the number of trial channels:

- RF channels can be masked out at configuration time
- In host-controlled operation, RF channel masking can be done dynamically by the host processor

The minimum number of RF channels is 6, corresponding to 2 trial channels. See section 3.4.5 for more details.

2.4.4 RF Coexistence Mechanisms

The two main mechanisms that allow a PurePath Wireless system to co-exist amicably in close proximity to other 2.4 GHz radio systems (including other PurePath Wireless networks) are:

- The adaptive frequency hopping scheme described in section 2.4.3 that ensures that RF channels used by other radio systems are avoided
- Listen-before-talk mechanism that measures energy in RF channel before transmitting and avoids transmitting if the channel is already in use

These mechanisms together ensure that other radio systems are minimally impacted by a PurePath Wireless audio network in normal circumstances. However, since a low-latency audio network by its very nature transports a very time-critical data stream, both mechanisms have adaptive thresholds to ensure that the audio network is given its fair share of RF spectrum in very crowded RF environments.

2.4.5 Antenna Diversity

Antenna diversity allows the PurePath Wireless network's protocol slaves to switch dynamically between two antennas to reduce effects of RF multipath fading. Using this feature requires an external antenna switch, which is controlled by the CC85xx. See section 2.1.9 for further details on the hardware aspects of antenna diversity.

The switching is done on a per-timeslot basis and the selected antenna is used for both master packet reception and subsequent slave packet transmission. Two different algorithms may be employed, depending on the protocol master's firmware version. Decision making is based on individual statistics for active RF channels, and on statistics from the nearest active RF channel for trial RF channels:

- Fast algorithm (selected automatically when connected to release 1.3.0 or later protocol master):
    - When an antenna diversity enabled protocol slave connects, the protocol master enables preamble extension of the master packet with 96 us, and the protocol slave enables its receiver 96 us earlier
    - At the start of the extended preamble, the protocol slave first tests the assumed best antenna 5 of 6 times and the currently assumed worst antenna 1 of 6 times. The result of this test will be compared with a dynamic performance threshold and the antenna used in the timeslot will be selected based on this comparison.
- Slow algorithm (selected automatically when connected to release 1.2.2 or earlier protocol master):
    - The protocol slave uses the currently assumed best antenna 5 of 6 times and the currently assumed worst antenna 1 of 6 times

The fast algorithm will adapt better to rapidly changing RF fading conditions, which is the typical case for portable equipment such as microphones, headphones and headsets.

2.4.6 Timeslot Alignment

Applications requiring more audio channels and/or protocol slaves than a single protocol master can provide, can achieve the goal by using multiple protocol masters with timeslot alignment.

If these protocol masters were to operate independently, without timeslot alignment, the slight differences in crystal frequency would cause their timeslot timing to drift slowly with respect to each other. This drifting would periodically cause degraded performance when one protocol master performing TX effectively jams the other protocol master performing RX and vice versa. When located in close proximity this jamming would occur even when the two networks operate on different RF channels.

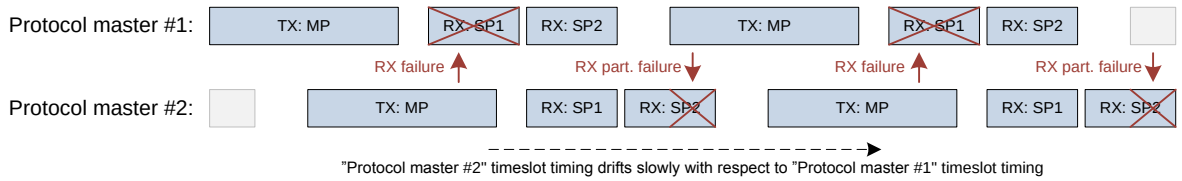


Figure 37 – Two masters without timeslot alignment

With timeslot alignment, a wire-based synchronization mechanism (using one GIO pin per device) ensures that the (non-USB) protocol masters perform TX simultaneously and RX simultaneously. One of the protocol masters functions as timing master (i.e. outputs the synchronization signal), while one or more protocol masters function as timing slaves (i.e. captures the synchronization signal).

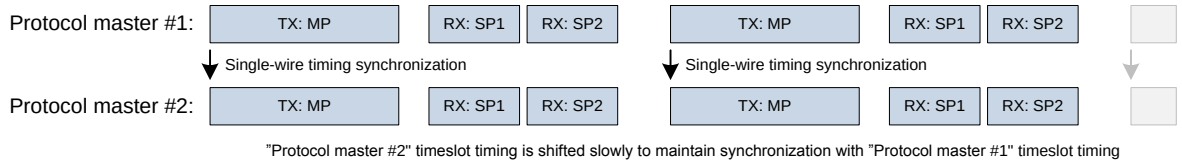


Figure 38 – Two timeslot aligned masters

As illustrated in Figure 38, the protocol masters are required to have identical timeslot timing. Refer to the PurePath Wireless Configurator's help system for information on relevant settings.

When timeslot-aligned protocol masters are not connected to any protocol slaves, an additional listen-before-talk mechanism is employed to avoid permanent collision on the active RF channels.

2.5 USB

The CC85x1 can implement a USB composite device, consisting of an audio device and a human interface device (HID).

The audio device handles all aspects of the audio streaming, including volume and sample rate control, and the human interface device handles remote control functionality.

2.5.1 Identification

USB devices are identified by a vendor ID and a product ID. These values and a device release number are specified at configuration time. The USB vendor and product IDs may not be selected at random:

- For prototyping, Texas Instruments' vendor ID 0x0451 may be used together with product IDs in the range 0x16BA - 0x16C1
- For released products, a purchased vendor ID must be used

Three USB string descriptors are implemented. These may be visible to the end-user on the USB host:

- Manufacturer name, specified at configuration time
- Product name, specified at configuration time
- Serial number, generated at run-time from the unique device ID (see section 2.2.1.1)

2.5.2 USB Audio Device

USB audio devices are described by typology block diagram, which allows the USB host to determine the structure and the capabilities of the audio device. This includes routing from input terminals to output terminals, through feature units (e.g. for volume control), mixer units etc. Two examples of such are shown below.

The PurePath Wireless Configurator's USB descriptor templates include audio device topologies for maximized operating system compatibility, so topology specification is in practice of no concern for most Configurator users.

The PurePath Wireless platform allows for both simple and complex audio device topologies, with the following restrictions:

- The USB input and output terminals must run at the same sample rate. Since the USB host controls sample rate per audio streaming interface, unidirectional audio streaming may support multiple sample rates, but bidirectional audio streaming can only support a fixed rate.
- One feature unit can be controlled for each audio streaming direction. For each of these feature units, the USB host control the mute and volume controls, but is limited to controlling either the master control **or** the individual channel controls (i.e. not both).
- Other functional units (FW 1.4.1 and later supports feature and mixer units) may be included for operating system compatibility, but these will only act as dummies, and have no function in the application.

The USB HT1 headphone topology (as defined by the Audio Device Class Definition for Basic Audio Devices 1.0) is shown in Figure 39. The block diagram shows a USB input terminal (ID1), a feature unit (ID2) for mute and volume control, and an output terminal (ID3) representing the headphone/speaker.

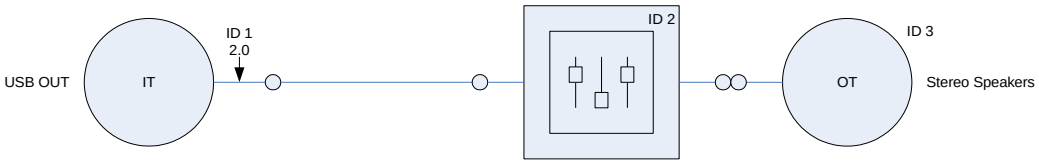


Figure 39 – USB HT1 headphone topology block diagram

The USB HS1 headset topology (as defined by the Audio Device Class Definition for Basic Audio Devices 1.0) is shown in Figure 40. This block diagram extends the previous one with a microphone input terminal, a USB output terminal, and three more functional units.

In this case, only feature units 2 and 5 will be controllable, while feature unit 7 and mixer unit 8, for mixing of the microphone input with the headphone output, are only "present" in the block diagram. If such functionality would be desired, it must be implemented statically in the protocol slave's audio device configuration.

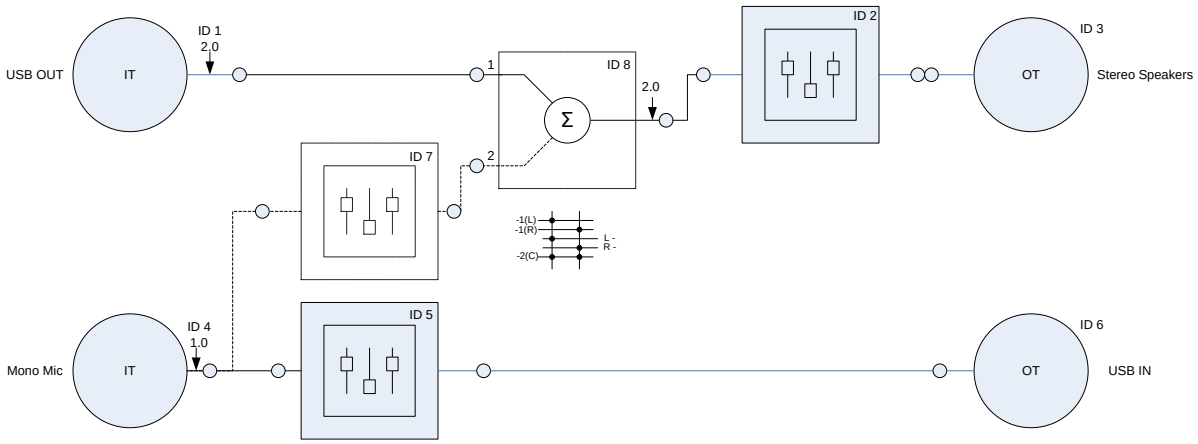


Figure 40 – USB HS 1 Headset Topology 1 Device Block Diagram

For further information on USB audio, refer to the following documents (available from www.usb.org):

- Audio Device Class Definition 1.0

- Audio Device Class Definition for Basic Audio Devices 1.0
- Audio Data Formats 1.0
- Audio Terminal Types 1.0

2.5.3 USB Human Interface Device

The USB human interface device is described to the USB host by a HID report descriptor, which defines the meaning and layout of the HID reports sent to the host when input is generated (e.g. keys are pressed and released).

A network can support either a pre-defined set of HID reports, including play control, US keyboard and mouse, or custom-defined reports based on user-written XML-based HID report definition files.

For further information on the USB HID class, refer to the following documents (available from www.usb.org):

- HID Device Class Definition 1.11
- HID Usage Tables 1.11

2.5.3.1 Pre-Defined HID Input Reports

FW 1.4.0 and later can support up to three different pre-defined HID reports, using endpoints EP3 IN and/or EP4 IN. The report input from the different protocol slaves is interpreted at the protocol master, and allows for HID input from multiple protocol slaves:

- Play control: Contains relevant audio controls from the HID usage table consumer page for volume and media player control
- US keyboard: A full, standard 101-key US English keyboard
    - Keyboard HID reports (when used) are interleaved with play control reports
- Mouse: Buttons and 8-bit horizontal (X) and vertical (Y) movement. This can be use to implement standard mouse or advanced pointing remote controls.
    - Mouse HID reports are sent on a separate endpoint to achieve stable non-interleaved report rate

2.5.3.2 Custom-Defined HID Input Reports

FW1.4.1 and later can alternatively use custom-defined USB HID reports, based on user-written XML-based HID report definition files. HID reports can be transmitted on endpoints EP3 IN and/or EP4 IN, with up to two different reports per endpoint. The HID report contents can in total (for all the reports) consist of up to 12 bytes.

The host-controlled protocol slave transmits raw HID report contents to the protocol master as remote control data. Since the HID reports are based on this user-specified, uninterpreted data, these networks can only support one protocol slave, and must employ pairing filtering on manufacturer ID.

The protocol master forwards the received HID report contents to the USB host, without interpretation of the data. The XML-based definition file specifies the HID descriptor(s) and HID report descriptor(s) sent to the USB host during enumeration, and the byte-array mapping from the remote control data to transmitted HID reports.

- HID reports are sent to the USB host upon content change and at the idle rate
- All HID report data is reset to zero upon protocol slave disconnection

The simple data transport mechanism only allows for certain types of HID report values:

- Single bits indicating whether a button is currently pressed, e.g. a play/pause button
- Multiple bits indicating the key code for a currently pressed key, e.g. on a standard keyboard

- Multiple bits indicating the value of a linear control, e.g. a joystick axis or a volume slider. Note that the automatic return to all-zero upon protocol slave disconnection must be handled appropriately, e.g. it should correspond to the neutral position of a joystick or minimum on a volume control slider.

The following types of HID report values are not recommended:

- Relative values, such as mouse movement, where input may be lost due to lack of timing control and data loss handling
- Analog values where the USB host calibrates the neutral position, e.g. a joystick where the neutral position is non-zero

Refer to the Configurator's help system for information on the HID report definition file format.

2.6 Power Management

The CC85xx supports the power states listed in Table 13. Refer to section 3.7.1 for further details on how manual and automatic mechanisms switch between these states.

Table 13 - Power states

State	Description
OFF	CC85xx and all external peripherals (if any) are turned off for minimum power consumption. For non-USB devices, the CC85xx can only be awakened by external asynchronous reset or by pulling the CSn pin low. For USB devices this state corresponds to USB suspend mode, and the CC85xx can only be awakened by USB resume signaling. USB remote wakeup is not supported.
ACTIVE	CC85xx and all external peripherals (if any) are turned on, and are fully active.
LOW-POWER (unsupported by USB)	Same as ACTIVE, except that the external audio device is limited to a low-power state where outputs shall be off. Typically entered when audio processing determines that all local audio outputs are silent. Using this state is optional.
LOCAL STANDBY (unsupported by USB)	Same as ACTIVE, except that the external audio device is limited to an inactive state where both inputs and outputs shall be off. Typically entered when not connected to a network. Using this state is optional.
NETWORK STANDBY (supported by FW 1.3.0 and later)	Same as ACTIVE, except that the external audio device is limited to a configurable state. If the audio device is powered down completely, CC85xx audio processing and audio streaming will also be deactivated, but network connection will still be maintained. Entering this state on the protocol master forces also the network's protocol slaves to enter the state (but with their own audio device state limit configuration).

3 Operation

3.1 Overview

The PurePath Wireless platform provides two different ways of operating the CC85xx; autonomous operation and host-controlled operation. For each device configuration the PurePath Wireless Configurator can also generate a firmware image for production test of the application hardware.

3.1.1 Autonomous Operation

In autonomous operation, CC85xx operates on its own without any need for supervision by a host processor. This reduces cost and time-to-market at the expense of flexibility.

The device is statically configured with parameters selected in the Configurator. Power, network and audio operations are controlled through a human user interface consisting of push-buttons, switches and status LED(s). The external audio device (e.g. an audio codec or digital input amplifier) is initialized and controlled by CC85xx over the I^2C interface. Battery voltage can be monitored for forced power-down. When powered down, network pairing information and relevant volume settings will be stored in non-volatile memory.

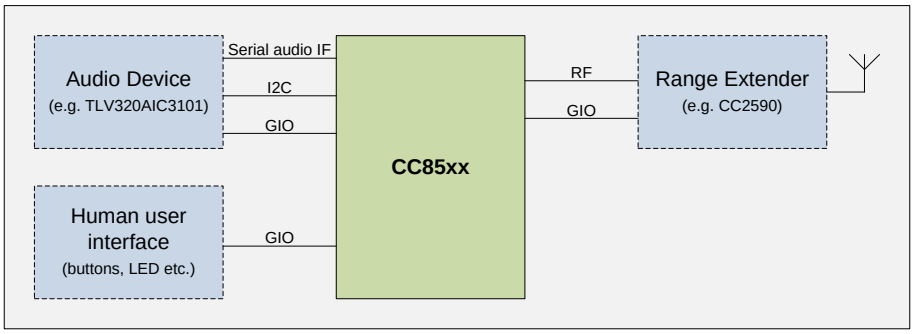


Figure 41 - Autonomously operated system

3.1.2 USB Device

CC85x1 operates on its own, and is controlled by the USB host it is attached to.

The device is statically configured with parameters selected in the Configurator. The USB audio device is controlled by audio device class requests, received from the USB host. The USB host can in turn be controlled by the end user, using the USB human interface device with the remote control features described in section 2.5.3.

Network pairing can be controlled through a human user interface consisting of a push-button for pairing, and a status LED. If a human user interface is undesired on the USB device (e.g. due to form factor), protocol slaves can use fixed or proximity-based pairing.

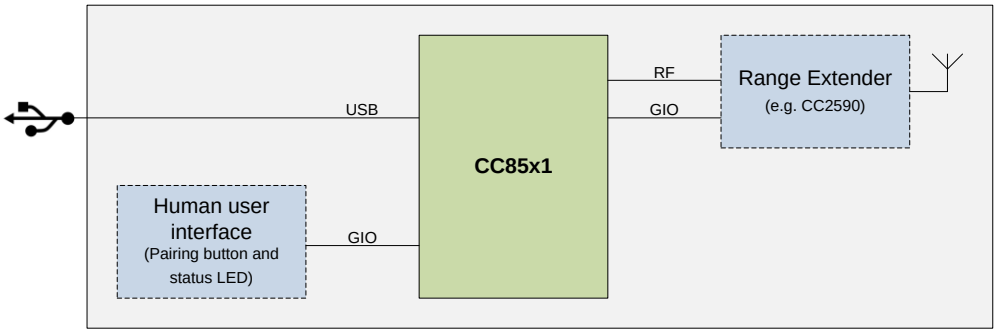


Figure 42 - USB Device

3.1.3 Host-Controlled Operation

In host-controlled operation, an external MCU or DSP operates the CC85xx through the SPI-based external host interface (EHIF), and controls most aspects of power management, network management, and audio-related operations (e.g. volume control). The CC85xx may or may not be responsible for controlling the external audio device (as configured in PurePath Wireless Configurator).

A low-bandwidth data side-channel is available for datagram transfers between the protocol master's host processor and the protocol slaves' host processors.

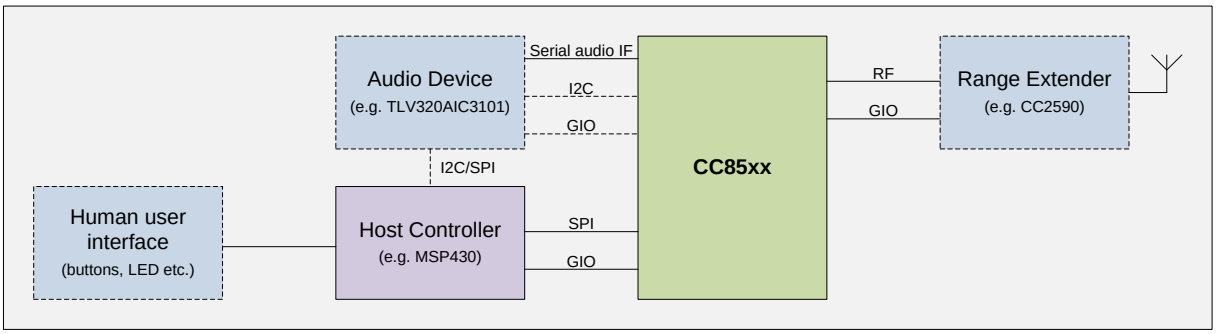


Figure 43 – Host-controller operated system

3.1.4 Production Test Firmware Generation

PurePath Wireless Configurator allows a production test firmware image to be output for any device configuration. The production test firmware is operated through the SPI-based external host interface, and allows for testing of connectivity, radio performance and radio regulatory compliance. The production test firmware adopts all peripheral setup from the corresponding application firmware, including support for the external audio device. This firmware does not support audio streaming, but provides instead a local EHIF-controlled tone generator and tone detector.

For USB devices, the production test image supports electrical verification of the USB data lines.

Since it is a separate firmware image, the production process will typically consist of these main steps:

- Production test image programming
- Production test execution
- Application image programming

Refer to EHIF command set reference in appendix B.3 for details.

For applications where space constraints excludes an SPI-based external host interface or for cost-constrained products, a simplified mini production test can be enabled. This allows for single GIO-driven, round-robin, transmitter test to output an un-modulated carrier at low, mid and high frequency.

Refer to the PurePath Wireless Configurator's help system for further details.

3.2 External Host Interface

The external host interface (EHIF) allows a host processor or similar to operate the CC85xx. The interface consists of a 4-wire SPI interface for EHIF command execution and a single GIO pin for level-triggered interrupt generation to the host processor. The SPI interface is described in detail in section 2.1.3.

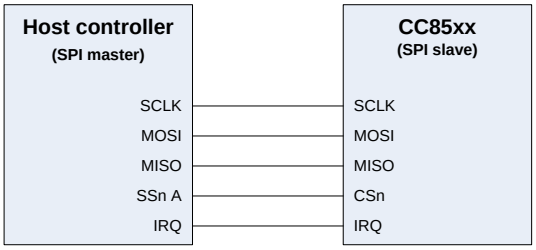


Figure 44 - EHIF overview

The EHIF command set is constructed from a set of basic SPI operations. A brief summary of these operations is provided in Table 14 and a detailed description is found in appendix B.1. Each EHIF operation consists of a CMD_REQ operation, which may be followed by a WRITE, READ or READBC operation.

Table 14 - Basic SPI operation summary

Operation – Description
`CMD_REQ(IN: command code, length of parameters, parameters; OUT: SPI status word)` **Command operation:** Each command has a unique command code. The different commands have different parameters, and some commands may have no parameters. The number of parameters bytes may therefore be zero and the parameter field may be omitted.
`WRITE(IN: length of data, data; OUT: SPI status word)` **Write operation:** Provides payload data for the EHIF command. The WRITE operation requires the host processor to provide the length of the data field.
`READ(IN: length of data; OUT: SPI status word, data)` **Read operation:** A command may use one of two read operations. The READ operation requires the host processor to provide the length of the data field, and can thus only be used when the number of output bytes is known in advance.
`READBC(OUT: SPI status word, length of data, data)` **Read operation:** A command may use one of two read operations. The READBC operation outputs the length of the data field, and can thus be used when the number of output bytes is not known in advance.

Due to internal processing, the host processor must wait until the external host interface is ready between each SPI operation. The external host interface signals that it is ready for the next operation by setting the MISO pin high when CSN pin is asserted.

3.2.1 Command Set Overview

The external host interface is available for all modes of operation (i.e. autonomous, host-controlled and production test), however each operational mode supports only a subset of the total EHIF command set:

- Autonomous operation: Device information, EHIF control, RF/audio performance statistics, utilities and calibration
- Host-controlled operation: Device information, EHIF control, audio network control/status, data side-channel, power management, volume control, remote control, RF/audio performance statistics, utilities and calibration

- Production test: Device information, EHIF control, RF test, audio test and IO test

Note that for autonomous operation, the external host interface should only be used for performance evaluation, testing and production. Furthermore, the interrupt pin is not available. To avoid interference with the power-toggle button and other button functions mapped to the CSn pin, the host processor/test controller must:

- Configure its SSn pin as open-drain
- Not use the SPI interface for communication with other devices than CC85xx
- Only assert CSn for <10 ms period of time, followed by >20 ms of inactivity

Table 26 in appendix B.3 lists all EHIF commands and indicates which commands are available for the different modes of operation.

3.2.2 Status Word and Interrupt Generation

A 16-bit status word is shifted out at the start of each SPI operation. The MSB of the status word (CMDREQ_RDY), which is output combinatorially on MISO as soon as CSn is pulled low, indicates whether or not EHIF is ready for new SPI operation. If MISO is low, the host processor can either wait for MISO to go high, or de-assert CSn and come back to check for it later.

Table 25 in appendix B.3 shows the contents of the SPI status word.

The external host interface interrupt line is asserted whenever one or more unmasked event flags are set. Some of the interrupt flags must be cleared manually using the EHC_EVT_CLR command; see Table 25 in appendix B.3 for details. The interrupt mask is set using the EHC_EVT_MASK command.

3.3 Human User Interface

A human user interface is available for autonomous operation and to some extent host-controlled operation, and allows the end-user to operate and monitor the CC85xx-based device through a button- and LED-based user interface. The buttons control power, network pairing, audio channel selection, remote control and volume, and the LED indicates the current network and power state. The detailed behavior of this user interface can be configured in the PurePath Wireless Configurator.

3.3.1 Button Input

The following button functions are supported:

- Power toggle (on/off)
- Network standby toggle (on/off)
- Network pairing on (off at timeout or success)
- Input and/or output volume increment, decrement and mute (on/off)
- Left/right output balance (protocol slaves only)
- Audio channel selection (cycle through defined channel sets)
- Remote control buttons (protocol slaves only), which may be mapped to functions such as play/pause, skip etc.

Each button can be configured in different modes, and the available modes depend on the nature of the button function. For instance, the network pairing button function is single-event (i.e. it makes no sense generating multiple events in a row), while the volume increment button is multi-event (i.e. it makes sense to generate multiple events in a row).

For remote control buttons, the protocol slave only associates a “function label” with each button, and transfers the button state. The protocol master implements the button action and defines event timing.

Table 15 - Button events

Event	Description	Supported button functions
CLICK	A single event is generated when the button is pressed and released within a specified interval.	All
HOLD	A single event is generated when the button has been pressed for a specified interval.	Power toggle Network pairing on Network standby toggle Input/output volume mute toggle Remote control
REPEAT	As long as the button is pressed, events are generated at a specified interval after a configurable delay.	Input/output volume increment/decrement Left/right volume balance control
CLICK+REPEAT	Combines the CLICK and REPEAT types, by generating a single event when the button is pressed, followed by multiple repeat events when the button is held for a configurable amount of time.	Input/output volume increment/decrement Left/right volume balance control
STATE	A remote control code is transmitted while the button is pressed.	Remote control

The active IO level (high or low) for each button is configurable. Relevant timing parameters are configurable, per button event type.

To reduce the total number of buttons, it is possible to map two functions to the same physical button. For instance:

- Volume increment (CLICK) and volume decrement (REPEAT) on the same physical button
- Power toggle (CLICK) and network pairing (HOLD) on the same physical button

STATE buttons functions cannot be combined with other events.

When CLICK or HOLD events are generated by remote control buttons, the protocol slave will emulate a 90 ms button push at the time the event occurs. See section 3.3.3 for further details on remote control.

3.3.1.1 Button Debouncing Algorithm

GIO pins connected to buttons are sampled every 10 milliseconds. The debouncing algorithm requires 8 identical samples to accept a state change:

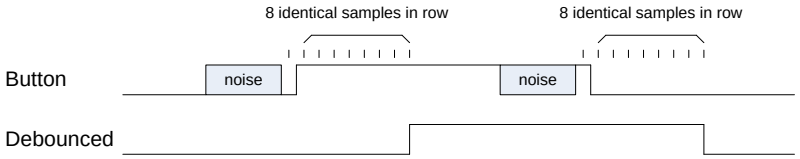


Figure 45 - Button debouncing

Also, to prevent the user from double-pushing a button and prevent similar effects when releasing a button, each button function has a minimum idle time between two pushes. If this idle time is violated, the second push will be ignored.

3.3.2 Network Status LED Output

The status LED is used to indicate the macro network state of the CC85xx device. The macro states are defined in Table 18.

For each network state, except OFF where the LED remains unlit, the PurePath Wireless Configurator can specify any LED blinking pattern, with a time resolution of 10 ms.

To save GIO pins, it is possible to share a physical GIO pin between an LED and button functions. The disadvantage is that the LED will always be lit while pushing the button.

3.3.2.1 Extended Status Indication

The status LED can also indicate low battery and which of two channels is used in automatic dynamic channel selection:

- Low battery can be indicated by an optional LOW BATTERY blinking pattern, and/or by switching to an alternate status LED. The low battery warning:
  - Starts when the measured battery voltage is below a configurable threshold 4 times in row
  - Ends when the measured battery voltage is above another configurable threshold 20 times in row
- Automatic dynamic audio channel selection can be indicated by switching to an alternate status LED. This option is available for mono microphone devices with two dynamic selection banks (e.g. “Microphone 0” and “Microphone 1”).

The alternate status LED can only be used for one of the above indications. It can be used to either change the color of the status LED (e.g. from green to red) or move the LED physically (e.g. from label “Channel 1” to label “Channel 2”).

When using the alternate status LED, a second GIO pin controls this LED in the same way as the first GIO pin controls the main status LED. The unused LED is always inactive.

3.3.3 Remote Control

Methods for production and consumption of remote control commands depend on the operation mode:

- Protocol slaves (producer):
  - Autonomous: Up to 8 buttons can be mapped to different remote control commands, mainly for play control (play/pause, volume up/down etc.)
  - Host-controlled: Can produce all pre-defined remote control commands, keys for a standard keyboard and a two-axis mouse, using the RC_SET_DATA EHIF command. The host processor controls the state of each key, i.e. whether it is currently pressed or not and keeps track of the mouse position. Alternatively, the host-processor can transmit raw data for custom-defined USB HID reports or proprietary remote control data for a host-controlled protocol master.
- Protocol masters (consumer):
  - Autonomous: GIO pins can indicate the current state of up to 8 pre-defined remote control commands, to allow for wireless button implementation. Volume control commands are automatically looped (with the protocol master's specified button timing) for remote volume control of all protocol slaves in the network. Network standby commands automatically affect the protocol master's power state, which in turn affects all protocol slaves' power state.
  - Host-controlled: Polls the remote control data from each protocol slave, using the RC_GET_DATA command (using the same format as on the producer)
  - USB device: Forwards automatically the received remote control data to the USB host through the human interface device. When using pre-defined HID input reports, network standby commands automatically affect the protocol master's power state (affecting the network status LED), which in turn affects all protocol slaves' power state.

The platform currently specifies the following proprietary set of pre-defined remote control commands:

Table 16 – Remote Control Commands

Command ID	Description	Autonomous handling/looping
1	Output volume mute	FW 1.2.0 or later required
2	Output volume increment	FW 1.2.0 or later required
3	Output volume increment	FW 1.2.0 or later required
4	Input volume mute	FW 1.2.0 or later required, not supported by USB-based protocol masters
5	Network standby enable	FW 1.3.0 or later required
6	Network standby disable	FW 1.3.0 or later required
7	Network standby toggle	FW 1.3.0 or later required
16	Play/Pause	FW 1.2.0 or later required
17	Scan next track	FW 1.2.0 or later required
18	Scan previous track	FW 1.2.0 or later required
19	Fast forward	FW 1.2.0 or later required
20	Rewind	FW 1.2.0 or later required

All other command IDs are currently reserved for future extensions.

3.3.3.1 Mouse Movement

Mouse movements are transferred using an unacknowledged protocol, where the protocol slaves convert movement to 16-bit horizontal (X) and vertical (Y) positions (with a random reference point), and the protocol master converts back to movement. This protocol prevents loss of movement.

- Host-controlled protocol slaves (producer):
  - Upon network connection, a random starting point (i.e. the 16-bit X and Y position values) is set using the RC_SET_DATA EHIF command (with EXT_SEL = 1). The host processor must keep track of the current position values.
  - When the mouse moves, the movement is added to the last set position, also using the RC_SET_DATA EHIF command (with EXT_SEL = 1). For example:
    - The last position set with was X=430, Y=24000
    - The mouse moves 6 pixels to the left (ΔX = -6) and 67 pixels down (ΔX = +67)
    - The new position becomes X=424, Y=24067
- Host-controlled protocol masters (consumer):
  - The host processor polls for mouse movements at regular intervals (no higher than 100 ms) using the RC_GET_DATA EHIF command. If the EXT_SEL field is 0, the mouse movement and mouse button data should be ignored.
  - The host processor must keep track of the current position values for each protocol slave. For each protocol slave, the starting point must be registered or re-registered:
    - When the protocol slave joins the network
    - If mouse reporting stops and then resumes (EXT_SEL goes from 0 to 1)
  - When the positions change, the movement is equal to the change since the last values read. By using 16-bit signed integer variables for the calculations, wrapping at 0xFFFF↔0x0000 can be ignored. For example:
    - The last position was X=15, Y=16700
    - The new position is X=-46, Y=16677
    - The mouse has moved 61 pixels to the left (ΔX = -61) and 23 pixels up (ΔX = -23)
- USB-based protocol masters (consumer):
  - Mouse reports are transmitted to the USB host automatically (every 8 ms as needed)
  - If movement is too fast to fit within the 8-bit signed values in the HID report, the movement is spread automatically over multiple reports (of N times ±127 and the remainder)

The mechanism relies on the movement speed being small compared to the 16-bit position space. To avoid underflow and overflow, which would cause a very large unexpected movement in the wrong direction, and to be compatible with USB, the host processor must limit/delay the input to the protocol slave as follows (for X and Y positions):

- At time t_0 the position is A
- At time t_0 + 100 ms the position has to be within the interval [A – 1587, A + 1587], the point of which is that
  - Over time the average movement speed should not exceed ±127 per 8 ms, but …
  - … faster movements back and forth within a limited position range are allowed

3.4 Network Functionality

This section describes how a PurePath Wireless device is identified in the network, how network establishment and maintenance is done, and how the data side-channel is operated.

3.4.1 Device Identification

Each PurePath Wireless device is identified by three 32-bit identification numbers:

Table 17 - ID numbers

State	Description
Device ID (DID)	Identifies uniquely each CC85xx device. The protocol master's device ID also serves as network ID, which protocol slaves use to join specific networks or re-join the previous network. The device ID is set in hardware by Texas Instruments during production test, and cannot be changed.
Manufacturer ID (MID)	Identifies uniquely the manufacturer of the CC85xx-based product. The value is used for identification during network scanning, as filtering criteria during network pairing (i.e. it can prevent network establishment by devices from other manufacturers), and can also be used to determine protocol compatibility for data side-channel communication. The manufacturer ID must be generated using the PurePath Wireless Configurator by reading out the device ID from a CC85xx that the manufacturer owns. Note that this device should be marked, and should not be passed on to other manufacturers. The manufacturer ID is statically configured in the device configuration.
Product ID (PID)	Identifies the CC85xx-based product. This value is used for identification during network scanning, as filtering criteria during network pairing together with manufacturer ID filtering (i.e. it can allow only specific products from the manufacturer), and can also be used to determine protocol compatibility for data side-channel communication. The product ID is entirely manufacturer defined. Each part of an application (e.g. headphone and base station) should preferably have a unique ID. The ID can also contain version numbering. The product ID is statically configured in the device configuration.

When a protocol slave joins a network, the protocol master can require the slave to have the same manufacturer ID as itself. If this criterion is not met, the protocol slave is not allowed to join the network.

Likewise, the protocol slave can apply filtering during its search for protocol masters, and only accept masters with same manufacturer ID as itself. The protocol slave can, when already filtering on manufacturer ID, also require the master's product ID to match the following criteria:

```
"Protocol master's PID" AND "PID mask" = "Reference PID" AND "PID mask"
```

The reference mask and reference product ID are configured statically in the PurePath Wireless Configurator for autonomous operation, and are given as parameters in the EHIF network commands for host-controlled operation.

3.4.2 Network Establishment

Initially, the audio network consists only of the protocol master, which periodically transmits information about its network. Protocol slaves are responsible for searching for and joining networks.

The network information broadcasted by the protocol master contains all relevant information, so that the protocol slaves can search for networks passively, that is without transmitting any packets and disturbing existing networks. The network information includes such as:

- Identification (Manufacturer ID, Network ID and Product ID)
- RF channel list for adaptive frequency hopping
- Pairing signal
- Streaming format (e.g. PCM16) and volume control information (for remote volume control) for each streamed audio channel
- List of all protocol slaves that are currently connected to the network
- Audio latency and sample rate

For a complete list of the scan results returned by the NWM_DO_SCAN command, refer to appendix B.3.3.

A “pairing” mechanism is used to associate protocol slaves with a protocol master, so that they can join and later re-join its network. The available pairing mechanisms for autonomous and host-controlled operation are outlined below. The mechanisms use the same underlying framework and are fully compatible across modes of operation.

3.4.2.1 Autonomous Operation, including USB Device

For autonomous operation, network establishment is based on pairing.

When powered up and not connected, a protocol slave will continuously search for the protocol master it was previously connected to, and, if found, attempt to join its network. Network pairing can be fixed (set during production) and/or be triggered by the end-user during active operation:

- Protocol slave: The end-user pushes the pairing button to search for a protocol master in close proximity (RSSI above configured threshold) and/or with pairing signaling enabled. The search will continue until a suitable master is found or until a configurable timeout expires. If unsuccessful, the slave will attempt to (re-)join the network it was previously connected to, if any.
- Protocol master (optional): The end-user pushes the pairing button to signalize that pairing is enabled. The pairing signal will remain on for a configurable interval or until the next protocol slave joins the network.

If the pairing is successful, the protocol slave stores the new network ID in non-volatile memory, and will thus be able to re-join the network after a power-down.

If the network is currently full, pairing will succeed (i.e. the network ID will be stored), however the protocol slave will not be able to join it until a slot is available.

3.4.2.2 Host-Controlled Operation

For host-controlled operation, network establishment can be based either on pairing or on network scanning, followed by joining a specific network found in the scan results.

The following EHIF commands are relevant for network establishment:

- Protocol master: NWM_CONTROL_ENABLE, NWM_CONTROL_SIGNAL, NWM_GET_STATUS
- Protocol slave: NWM_DO_SCAN, NWM_DO_JOIN, NWM_GET_STATUS, NVS_SET_DATA, NVS_GET_DATA

On a host-controlled protocol master, the NWM_CONTROL_ENABLE command enables and disables the PurePath Wireless network. When disabled, the protocol master is inactive, and no network can be

established or maintained. When enabled, the protocol master will start broadcasting its network information and allow protocol slaves to join it.

After enabling the network on the protocol master, network pairing is accomplished as follows:

- Protocol slave: The NWM_DO_SCAN command is used to search for a protocol master in close proximity (RSSI above specified threshold) and/or with pairing signaling enabled. If a suitable protocol master is found before the specified timeout expires, its network information will be returned. This network is then joined using the NWM_DO_JOIN command with the returned network ID.
- Protocol master (optional): The pairing signal is set using the NWM_CONTROL_SIGNAL command. The pairing signal can also be cleared using the same command, or it will be cleared automatically when the next protocol slave joins the network.

Alternatively, the protocol slave can scan for multiple networks using the NWM_DO_SCAN command, and use more details in the returned information to select which one to join.

The protocol slave's host processor is responsible for providing the network ID when re-joining a network, and is hence also responsible for storing it during any power-down periods. The network ID can be read out from the CC85xx while it is connected to the network, using the NWM_GET_STATUS command, or is already known from scan results (after NWM_DO_SCAN). The NVS_SET_DATA and NVS_GET_DATA commands can be used for network ID storage in order to save EEPROM cost.

The following flow diagram illustrates how the EHIF command set is used on a protocol slave to find and join a network:

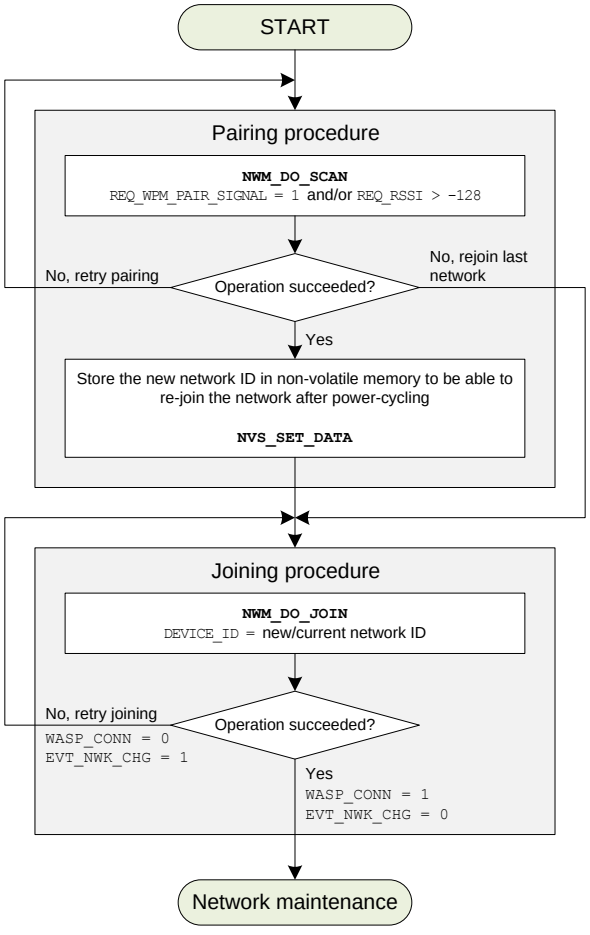


Figure 46 - Network establishment flow for protocol slaves

To re-join an old network, the host processor uses the NWM_DO_JOIN command with the old network ID as input parameter.

3.4.3 Network Maintenance

A protocol master is said to maintain a network when it regularly broadcasts network information. It will do this regardless of the number of connected protocol slaves.

A protocol slave is said to maintain a network when it is connected to a network and regularly receives packets from the protocol master. The connection is lost when it does not receive any packets over a period of about 300 ms (for the nominal timeslot period of 2.5 ms).

No particular action is required to leave a network, and unexpected loss of power supply on a protocol slave will not cause any problems. The protocol slave simply ceases all communication with the protocol master, and is then removed from the protocol slave list in the network information and ejected from the network.

The status LED(s) described in section 3.3.2 can be used to indicate the current network state:

Table 18 - Macro network states/status LED patterns

State/LED pattern	Description
OFF	The device is powered down.
ALONE	Protocol master: The device is not connected to any protocol slaves. Protocol slave: The device is not connected to a protocol master, and is searching for the network it was previously connected to, if any.
STANDBY	The device is connected, but is not streaming audio. This state occurs when the protocol master has no valid audio clock or no valid digital audio input.
ACTIVE	The device is connected and is streaming audio.
PAIRING	Overrides the ALONE, STANDBY and ACTIVE LED patterns while pairing is attempted and also otherwise while NWM_DO_SCAN is running.
LOW BATTERY	Overrides the ALONE, STANDBY, ACTIVE and PAIRING LED patterns when to warn about low battery voltage.

3.4.3.1 Autonomous Operation, including USB Device

In autonomous operation, the network is maintained automatically with no need for interaction by the end-user. If a protocol slave loses network connection, it will attempt to re-join the network indefinitely or until the end-user powers the device down.

3.4.3.2 Host-Controlled Operation

The following EHIF commands are relevant for network maintenance:

- Protocol master: NWM_CONTROL_ENABLE, NWM_GET_STATUS
- Protocol slave: NWM_DO_JOIN, NWM_GET_STATUS
- NVS_SET_DATA and NVS_GET_DATA can be used to store and retrieve information

A protocol master will maintain the network automatically until it is disabled using the NWM_CONTROL_ENABLE command. When a protocol slave joins or drops out of the network, an EVT_NWK_CHG event (with possible interrupt generation) will be generated, and the updated list of protocol slaves can be read out using the NWM_GET_STATUS command.

A protocol slave will, once connected, stay connected until it leaves the network actively using the NWM_DO_JOIN command (with 0x00000000 as network ID), or loses the connection to the protocol master due to poor RF conditions. The EVT_NWK_CHG event (with possible interrupt generation) indicates that the

NWM_DO_JOIN command has failed, or that the protocol slave has been disconnected after a successful NWM_DO_JOIN command.

3.4.4 Data Side-Channel

The data side-channel is only available in host-controlled operation

The data side-channel is an ordered, assured delivery, integrity-checked datagram service. It allows the host processor to send and receive datagrams with 1-32 bytes of user-defined payload.

Transfer bandwidth is not assured, and each network device can only transmit and receive one datagram per network timeslot. This translates to a best-case data throughput of 100 kbps out of each device with a 2.5 ms timeslot period (when using maximum length datagrams). When multiple protocol slaves supporting the data side-channel are connected, the best-case throughput is effectively divided by the number of protocol slaves.

The following EHIF commands are relevant for data side-channel communication:

- Protocol master and slave: NWM_GET_STATUS, DSC_TX_DATAGRAM, DSC_RX_DATAGRAM

For each protocol slave a bi-directional data side-channel connection with the protocol master can be established. The protocol master can send datagrams to and receive datagrams from each protocol slave, while a protocol slave can only send datagrams to and receive datagrams from the protocol master.

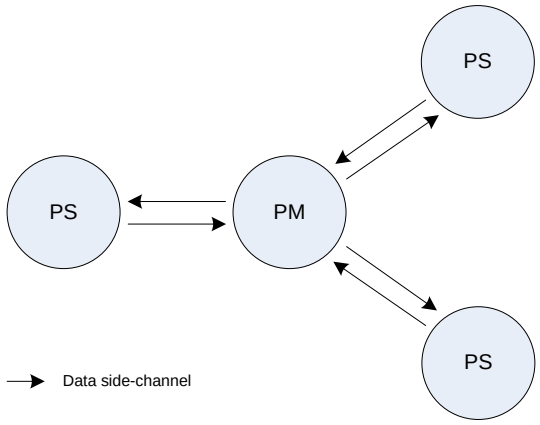


Figure 47 – Data side-channel datagram flow

A datagram consist of a connection reset flag used to establish a data side-channel connection, the unique device ID of the datagram's intended recipient (also referred to as address field), and 1 to 32 bytes of user-defined payload data:

Connection reset flag	Recipient device ID	1 – 32 bytes of payload

Figure 48 - Datagram contents

The datagram payload contents are entirely user defined, and it is up to each host processor to determine whether or not the other host processor uses a compatible payload format and communications protocol. The method for determining such compatibility is described in the next section.

If a network member is autonomously operated, it will not transmit any datagrams and discard any received datagrams.

3.4.4.1 Connection Establishment and Loss

Data side-channel connections must be established independently for each data direction. It is possible to use only one-way communication and leave one data direction (e.g. from protocol slave to protocol master) unused.

The NWM_GET_STATUS command returns information about whether connected devices support the data side-channel or not. Protocol slaves can also use NWM_DO_SCAN prior to joining a network to find networks supporting data side-channel communication.

A datagram transmitter must ensure that the recipient is compatible, and must use a recipient's manufacturer ID and product ID to determine this. If compatible, the transmitter establishes the connection by sending the first datagram with the connection reset flag set, using the DSC_TX_DATAGRAM command.

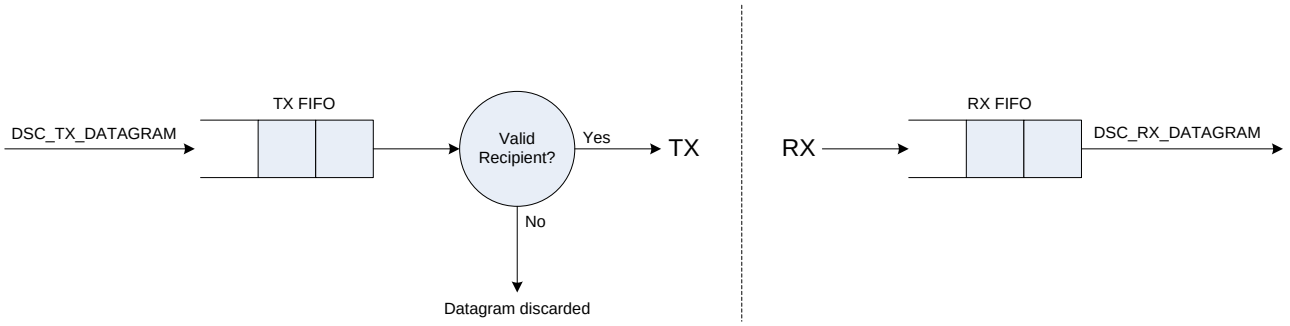
An established data side-channel connection (pair) is closed down whenever the corresponding network connection is lost. Such disconnection events are signalized through the EVT_NWK_CHG flag. The status of a data side-channel connection is checked every time a datagram is next in line to be transmitted. The queued datagram is silently dropped if the data side-channel is not established or has for some reason closed down. The affected data side-channel connections must then be re-established when/if the device is re-connected.

The connection reset flag is used by the transmitter to initialize communication when (re-)establishing a connection. On the recipient, the EVT_DSC_RESET flag in the EHIF status word signalizes that a datagram with the connection reset flag has been received. The flag is also present in the datagram read out using the DSC_RX_DATAGRAM command, and indicates that the connection is being (re-)established and that any previous unfinished communication has been terminated.

Note that the connection reset flag can be set in every datagram without error, but the host processor should make sure that this only happens in the establishment of the data side-channel or to explicitly reset communication.

3.4.4.2 Queuing Mechanisms

Each device has one RX queue and one TX queue. Each queue can contain 0 to 3 datagrams, and are implemented as FIFOs (first in first out).



3.4.4.2.1 Writing Datagrams to the TX Queue

Datagrams are inserted to the TX queue using the DSC_TX_DATAGRAM command. The datagram's destination is specified with the recipient's device ID, and this ID must be associated with an established connection unless the reset flag is enabled. Otherwise, the datagram will be dropped silently. The datagram will also be dropped silently if the device is not connected to the recipient.

The EVT_DSC_TX_AVAIL flag is set whenever there is available space in the TX queue. The EVT_SPI_ERR flag should be checked after the DSC_TX_DATAGRAM command to see if the datagram was successfully inserted.

Use the following steps to insert a datagram to the TX queue:

- Make sure that the EVT_DSC_TX_AVAIL flag is set and that the EXT_SPI_ERR flag is cleared before proceeding. Otherwise, abort the DSC_TX_DATAGRAM command after the EHIF status word, and clear the EVT_SPI_ERR flag using the EHC_EVT_CLR command.

- Insert the datagram into the TX queue using the DSC_TX_DATAGRAM command

3.4.4.2.2 Reading Datagrams from the RX Queue

Datagrams are read from the RX queue with the DSC_RX_DATAGRAM command. The command reads out one datagram with payload, the connection reset flag and the address field from the queue, provided that the queue is not empty.

The address field in datagrams transmitted from protocol slaves to protocol master is changed from the master's device ID to the slave's device ID. This allows the protocol master to identify the source of the datagram. The field is unchanged for datagrams transmitted from the protocol master to the protocol slaves.

The EVT_DSC_RX_AVAIL flag is set whenever there are unread datagrams in the RX queue. A datagram receiver is responsible for reading out received datagrams as quickly as possible. Otherwise, the RX queue will fill up, and eventually block the datagram transmitter's TX queue.

Use the following steps to read a datagram from the RX queue.

- Check that the EVT_DSC_RX_AVAIL flag is set
- Read out a datagram with the DSC_RX_DATAGRAM command
    - If the number of bytes returned by the READBC command is 5, and the address field is 0x00000000, the RX queue was empty when the command executed. This can happen if datagrams are read out in rapid succession (due to delay in clearing the EVT_DSC_RX_AVAIL flag).
- Handle the value of the reset flag appropriately

3.4.4.3 Responsibilities of the Host Processor

The CC85xx ensures an ordered, assured delivery and integrity checked data side-channel. This simple mechanism places few restrictions on the host processors using the datagram service, and they do play an important role in making data side-channel communication run smoothly and without problems. This section summarizes the responsibilities of the host processor:

- Establish (and re-establish if connection is lost) the data side-channel by setting the connection reset flag in the first datagram
    - Datagrams without an established connection are silently dropped when next in line to be transmitted
    - The connection reset flag can be set in any datagram without error, but the host processor should make sure that this only happens in the first datagram after connecting to a network
    - A datagram with the reset flag set should always be interpreted as the first datagram after a connection drop-out or power-cycling at the transmitter.
- Datagrams can only be sent to devices that support the data side-channel, and only after making sure that the recipient is compatible
    - Sending datagrams to devices that does not support the data side-channel will result in the datagram to be ignored (i.e. these datagrams will be dropped silently)
    - Datagrams bound to unconnected devices can be inserted to the TX queue, but these will be silently dropped when next in line to be transmitted
- Datagrams must be read out quickly of the RX queue when received.
    - The TX queue at the transmitter will be stalled if the RX queue of the first recipient is full
- Implement additional integrity checking (e.g. CRC32 or arithmetic checksum) for critical data transfers, such as firmware upgrade over-the-air. The basic integrity checking provided (based on a combination of CRC16 and CRC8) will detect most errors; however certain bit-inversion patterns will result in a false positive integrity check.

3.4.5 Manual Frequency Planning

High duty-cycle RF interferers that operate at known channels can be avoided with manual frequency planning. It is possible to remove up to 12 of the 18 RF channels for a minimum of 6 channels (with 4 as active and 2 as trial). There are two options:

- If the interferer's frequency usage is known and static, the CC85XX's overlapping and nearby RF channels can be masked out at configuration time
- If the interferer's frequency usage is unknown at configuration time, but is static or rarely moves during operation, RF channel masking can be done dynamically during operation, using the NWM_SET_RF_CH_MASK EHIF command. This command is available both in host-controlled and autonomous operation.

When changing the RF channel mask dynamically, the transition to the new channel set happens over time to minimize risk of audio drop-outs or loss of network connection.

Note that excluding RF channels may increase the effect of RF multipath fading, reducing audio streaming and network link robustness. It is therefore generally not recommended to exclude RF channels unless absolutely needed.

3.5 Audio Functionality

This section describes how all audio related functionality is operated, including audio stream control (i.e. routing of audio channels on the producer and the consumer), sample rate control, external audio device control and volume control.

3.5.1 Audio Streaming Control

Audio channel routing ensures that each audio channel input on the audio producer's external audio interface is transferred to the correct audio channel output on the audio consumer's external audio interface.

Refer to section 2.3.1 for a detailed description of the audio channel routing concept.

The serial interface format is statically configured. Available options are described in section 2.1.5.

3.5.1.1 Static Logical Channel Mapping

Static logical channel mapping is used by all protocol masters and by most autonomously operated protocol slaves.

For each streamed audio channel, the configuration establishes a connection between:

- On the external audio interface side: Audio I/O pin (AD0, AD1 or AD2) and an I^2S(/LJF/RJF) or DSP channel on that pin
- On the wireless audio streaming side: The logical channel ID that identifies the audio channel on the PurePath Wireless network

An example of a static logical channel mapping for a protocol master is given below. Note that this mapping also specifies the streaming format to be used. Protocol slaves do not specify any streaming format, but uses the format used by the protocol master

Table 19 - Static logical channel mapping

Logical channel	Audio I/O	I²S channel	Streaming format
Front left	AD0	LEFT	PCM16
Front right	AD0	RIGHT	PCM16

3.5.1.2 Dynamic Channel Selection for Autonomous Operation

Dynamic channel selection is optional, and can be used by autonomously operated protocol slaves supporting one and in some cases two audio channels.

Switching can be done while in active operation, and causes only a short audio drop-out. Autonomous operation provides three selection methods:

- A button, which allows the end-user to cycle through 2 to 8 logical channel banks
  - If using the power button function to turn the device on and off, the state of the dynamic channel selection button will be stored during power-down
- 1, 2 or 3 GIO pins to select between 2, 4 or 8 logical channel banks, respectively. The manufacturer can:
  - Hardwire the GIO pins to select logical channel
  - Add a DIP-switch or similar to let the end-user select logical channel
- Automatic channel selection, which allows mono microphones to select the first unused microphone logical channel (if any) without need for user intervention
  - When the dynamic selection count is 2, the current selection can be indicated using the alternate network status LED described in section 3.3.2.1

For the streamed audio channel(s), the configuration establishes a connection between:

- On the external audio interface side: Audio I/O pin (AD0, AD1 or AD2) and an I^2S(/LJF/RJF) or DSP channel on that pin
- On the wireless audio streaming side: The logical channel ID(s) that identify the selected audio channel(s) on the PurePath Wireless network

An example of a dynamic channel selection mapping is given below:

Table 20 – Dynamic channel selection logical channel mapping (using 2 GIO pins)

Dynamic selection	Logical channel	Audio I/O	I²S channel
0 (0b00)	Front/primary left	AD1	LEFT+RIGHT
1 (0b01)	Front/primary right		
2 (0b10)	Side left		
3 (0b11)	Side right		

3.5.1.3 Dynamic Channel Mapping for Host-Controlled Operation

Host-controlled protocol slaves specify the logical channel mapping after joining a network, using the NWM_ACH_SET_USAGE command.

For each streamed audio channel, the configuration establishes a connection between:

- On the external audio interface side: Audio I/O pin (AD0, AD1 or AD2) and an I^2S(/LJF/RJF) or DSP channel on that pin
- On the wireless audio streaming side: An index number used by the NWM_ACH_SET_USAGE command to establish the logical channel mapping

An example of a dynamic channel mapping is given below:

Table 21 - Dynamic logical channel mapping

Audio I/O	I²S channel	NWM_ACH_SET_USAGE index
AD1	LEFT	0
AD1	RIGHT	1

The NWM_ACH_SET_USAGE command takes a 16-byte array as input, with one byte for each logical channel ID. To map a serial audio interface channel to a logical mapping, its “NWM_ACH_SET_USAGE index” should be entered at the position of the desired logical channel ID, and all unmapped logical channel IDs should be set to 0xFF. For example (using the configuration in Table 21):

- To map AD1/LEFT to “Front left” and AD1/RIGHT to “Front right”, the byte array becomes

  `00,01,FF,FF;FF,FF,FF,FF;FF,FF,FF,FF;FF,FF,FF,FF`

- To map AD1/LEFT to “Subwoofer” and AD1/RIGHT to “Front center”, the byte array becomes

  `FF,FF,FF,FF;01,00,FF,FF;FF,FF,FF,FF;FF,FF,FF,FF`

3.5.2 Sample Rate Control

When using an external audio clock source on the protocol master, it will detect and automatically switch between the sample rates it is configured to support. If an unsupported sample rate is used by the external clock source, the audio streaming will be halted, and all outputs will be silent.

When using the internal audio clock source on the protocol master, it will use the configured default sample rate.

For USB-based protocol masters with multiple-sample rate configuration, the sample rate is controlled by the USB host, and the network switches automatically.

All protocol slaves will adapt to the protocol master's sample rate, and will do so dynamically. If the protocol master switches sample rate, the protocol slave will quickly shut down and restart its audio processing and external audio device at the new sample rate – normally within less than the audio latency period.

After power-up, the protocol slave will by default generate audio clock at the highest supported sample rate. If the sample rate used by the protocol master is unsupported by the protocol slave, the protocol slave's audio clock will shut down and remain so until the protocol slave is reset or the protocol master switches back to a supported sample rate.

3.5.3 External Audio Device Control

The CC85xx may handle the external audio device automatically. This is a requirement for autonomous operation, but is optional for host-controlled operation.

PurePath Wireless Configurator versions 1.1.1 and later define all properties of the supported external audio devices using an XML-based file format. Refer to the Configurator's help system for details on creating and modifying audio device definition files.

The CC85xx initializes, controls and monitors the external audio device by a combination of GIO pins, the I^2C interface or both:

- I^2C interface
    - Configures the external audio device through a series of I^2C configuration sequences, which take the audio device between different activity levels, ranging from OFF (where the device is powered down as much as possible) to ACTIVE (where the device is fully active). These configuration sequences can be customized in the PurePath Wireless Configurator.
    - Controls input and output volume

      Note that the PurePath Wireless Configurator provides a short description of each audio device, including the contents of the default configuration sequences, and which registers are used for volume control.
- Control and status pin interface
    - Up to 4 GIO pins can be used to control the audio device and associated circuitry
        - Can be driven or pulled high or low during the audio device state transitions
        - The available pin functions are determined by the audio device definition file
        - The GIO pins to be used are configured in the PurePath Wireless Configurator
    - Digital audio input valid pin
        - Signalizes to CC85xx on a GIO pin whether the input from a digital receiver is valid or not. This prevents CC85xx from treating Dolby Digital and DTS streams as PCM sample streams.
        - The GIO pin to be used is configured in the PurePath Wireless Configurator
        - The usage of this pin is optional (in FW1.0.1 and later). The pin function is disabled by default, and can be enabled through custom setup of the external audio device.

3.5.3.1 State Configuration Sequences

The external audio device will be in one of five states:

Table 22 - Audio device states

State	Description
ACTIVE	All inputs (if any) and outputs (if any) fully active.
LOW-POWER	All inputs (if any) are fully active; all outputs (if any) are in low-power mode.
INACTIVE	Configured and powered up, all inputs (if any) and outputs (if any) are in low-power mode.
SR-SWITCH	All inputs (if any) and outputs (if any) are prepared for sample rate switching or audio clock being enable/disabled at power-up/power-down.
OFF	Unconfigured and powered down for minimum power consumption.

To transition between each the different audio device states, the state configuration sequences shown in Figure 49 must be defined:

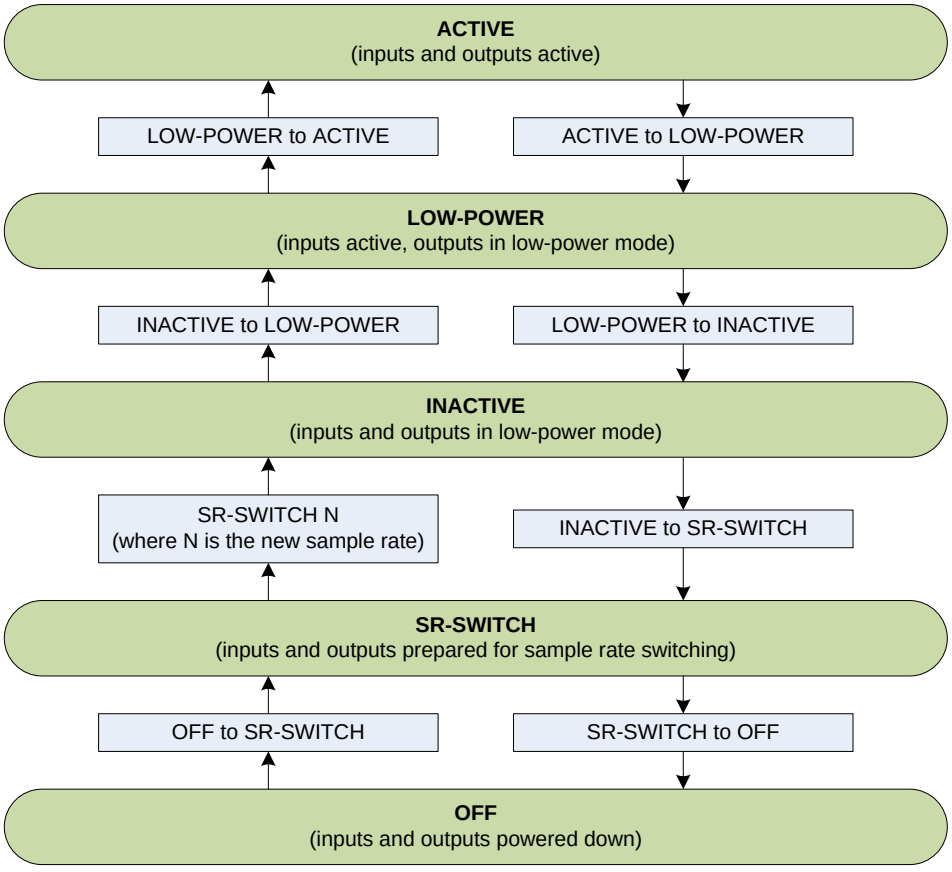


Figure 49 - Audio device state transitions

The configuration syntax used by the PurePath Wireless Configurator is compatible with other tools provided by Texas Instruments (e.g. the “AIC310x” configuration tool). In addition, the syntax has been extended to provide support for controlling up to 4 GIO pins. Refer to the Configurator’s help system for a complete description of the syntax.

Some audio device register operations may depend on a reliably running audio clock. The CC85xx's internal audio clock source will (when used):

- Never run during transitions between the OFF and SR-SWITCH states
- Always run during transitions between the SR-SWITCH, INACTIVE, LOW-POWER and ACTIVE states

Sample rate dependencies can be specified for audio devices that require it. There is one such configuration sequence for each supported sample rate.

3.5.4 Audio Volume Control

There is built-in support for both local and remote volume control:

A protocol master can:

- Control its own input and output volume, AND/OR
- Control the input and output volume of all connected protocol slaves

A protocol slave can:

- Control its own input and output volume, OR
- Let the protocol master control the input and output volume

The actual volume control takes place in the external audio device. When using an audio device with I^2C-based configuration and integrated support for volume control (e.g. the TLV320AIC3101 audio codec), the local input and output volume settings will be communicated to the device.

When the host processor is responsible for operating the external audio device, the volume settings can be read through the external host interface.

Each network node supports locally one input volume setting and one output volume setting. In addition, host-controlled nodes can specify individual logical channel volume offsets (e.g. for speaker system calibration), and autonomous protocol slaves can implement left/right output balance control.

The volume control resolution is 0.125 dB. For external audio devices with lower resolution, the settings will be truncated when being communicated to the device. To harmonize all volume settings throughout the network, each volume setting goes from a configurable (negative) minimum value to 0.0 dB. Any volume setting between -128.0 dB and the minimum setting will result in muting.

A configurable fixed offset is added to each volume setting before it is communicated to the external audio device. This offset corresponds to the gain applied in the I^2C volume control registers when the volume setting is at maximum (0.0 dB). The minimum volume setting is limited by this fixed offset and the range of the audio device' registers.

3.5.4.1 Autonomous Operation

In autonomous operation, volume control is based on button input. Depending on audio streaming direction(s) in the application, an arbitrary subset of following buttons can be used:

- Independent increment, decrement and mute toggle buttons for
    - Local input volume
    - Local output volume
    - Remote input volume (protocol master only, controlling local inputs on protocol slaves)
    - Remote output volume (protocol master only, controlling local outputs on protocol slaves)
- Left/right balance buttons for local stereo output (protocol slaves only)
    - Pressing both buttons resets the balance

The volume increment/decrement amount per button event and the rate of button events are specified at configuration time.

When the autonomous protocol master controls the input and/or output volume of the network's protocol slaves (autonomous or host-controlled), the protocol slaves can indirectly affect these volume settings by using remote control commands. This is done by an extra, always present, set of emulated volume buttons that the protocol slaves can "push" by activating the correct pre-defined volume remote control commands. This requires a FW 1.2.0 or later protocol master.

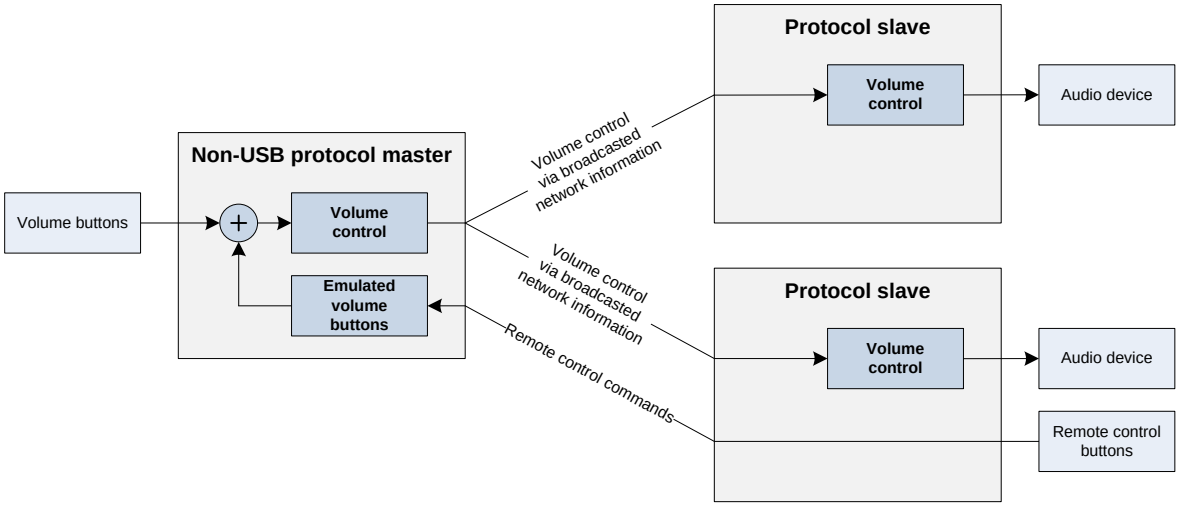


Figure 50 – A protocol slave using remote control commands to control volume for all slaves

When using a power toggle button for power control, the CC85xx will be able to perform a clean power-down of the system, and store relevant volume settings to non-volatile memory. These settings will be restored at power-up. A configurable default volume setting is used at the first power-up when saving the volume settings, and at every power-up when not saving the volume settings.

3.5.4.2 USB Devices

For USB-based protocol masters, volume and mute control class requests to the audio device topology feature units (see section 2.5) affect remote volume control of the protocol slaves. The USB host's volume settings can in turn be controlled by protocol slaves through the USB human interface device.

Note that protocol slaves configured with local volume control will ignore the volume settings communicated by the USB host. Also note that the input volume muting remote control command is unsupported by the currently available pre-defined USB HID reports, and will have no effect when using a USB-based protocol master.

3.5.4.3 Host-Controlled Operation

The following EHIF commands are relevant for volume control:

- Protocol masters and slaves: VC_SET_VOLUME, VC_GET_VOLUME, NVS_SET_DATA, NVS_GET_DATA

The VC_SET_VOLUME command allows the host processor to perform the following operations, in a local or remote (protocol masters only) scope:

- Mute, unmute and mute toggle
- Change volume setting by absolute value (between -128.0 dB and 0.0 dB), or by relative value (between -128.0 dB and +127.875 dB)
- Offset the individual logical channels at the input or output

For host-controlled protocol masters, the host processor is responsible for translating remote control input, polled by RC_GET_DATA, into volume changes.

The VC_GET_VOLUME command allows the protocol master to read the current local and remote input and output volume settings, and allows the protocol slave to read the current local input and output volume settings. The EVT_VOL_CHG event is generated whenever the local input or output volume setting changes.

3.5.4.4 Smooth Volume Changes

When incrementing or decrementing volume settings, for instance in steps of 2 or 3 dB, an internal mechanism will make the change seem smooth by applying it in smaller steps over a configurable period of time.

The VC_GET_VOLUME command reads the output of the smoothening mechanism, and an increment or decrement will result in a series of EVT_VOL_CHG events being generated at a 10 ms interval.

For remote volume control, changes happen instantly at the protocol master and smoothly at the protocol slave.

3.6 USB Functionality

This section describes operation of the CC85x1 protocol master by the USB host. This information is only relevant to understand observations on the USB bus and for development of custom USB drivers.

3.6.1 Supported Control Requests

This section is only relevant when writing custom audio device drivers for the CC85x1. Normal usage of the CC85x1 should not require this.

Table 23 – Supported standard and class requests – all other requests return STALL

State	Request	Description
Standard	GET_STATUS	Standard behavior, valid recipients: DEVICE, ENDPOINT
	CLEAR_FEATURE	Standard behavior, valid recipients: ENDPOINT
	SET_FEATURE	Standard behavior, valid recipients: ENDPOINT
	SET_ADDRESS	Standard behavior
	GET_DESCRIPTOR	Standard behavior
	GET_CONFIGURATION	Standard behavior
	SET_CONFIGURATION	Standard behavior
	GET_INTERFACE	Standard behavior
	SET_INTERFACE	Standard behavior
Audio class	SET_CUR	Set attribute value. Valid recipients are supported feature unit controls (MUTE and VOLUME) and sample rate control (if supported).
	GET_CUR	Get current attribute value. Valid recipients are supported feature unit controls (MUTE and VOLUME) and sample rate control (if supported).
	GET_MIN	Get minimum attribute value. Valid recipients are supported feature unit controls (MUTE and VOLUME) and sample rate control (if supported).
	GET_MAX	Get maximum attribute value. Valid recipients are supported feature unit controls (MUTE and VOLUME) and sample rate control (if supported).
	GET_RES	Get attribute resolution. Valid recipients are supported feature unit controls (MUTE and VOLUME).
HID class	SET_IDLE	Set idle duration for one or all HID reports.
	GET_IDLE	Get idle duration for one or all HID reports.

3.6.2 Audio Streaming Functionality

The CC85x1 supports two endpoints for audio streaming

- EP1 is an isochronous OUT endpoint for transfer of audio samples from the USB host. The endpoint can be configured as adaptive or synchronous (if EP2 also is synchronous).
- EP2 is an isochronous IN endpoint for transfer of audio samples to the USB host. The endpoint can be configured as asynchronous or synchronous (if EP1 also is synchronous).

To maintain audio latency and undisturbed wireless clock distribution, CC85x1 implements the following error handling:

- Missing start-of-frame
- EP1:
    - CRC errors are ignored.
    - Missing packets and packets with incorrect length are handled by discarding excessive data and inserting silence for missing data. The correction mechanism operates with an error margin of +/- 2 whole samples from the expected number of samples. For other received lengths, CC85x1 keeps track of and uses the expected number of samples (including the (9 x 44) + (1 x 45) sequence for 44.1 kHz sample rate).
- EP2:
    - Missing IN tokens are handled.

3.6.3 Human Interface Device Functionality

CC85x1 supports two endpoints for HID reporting:

- EP3 is an interrupt IN endpoint, with poll interval specified by the USB descriptor template
- EP4 is an interrupt IN endpoint, with poll interval specified by the USB descriptor template

HID reports are sent upon content changes, and then periodically at the specified idle duration. The default idle reporting period is 500 ms, and can be changed by the SET_IDLE class request. Idle duration 0 disables idle reporting.

For layout of the pre-defined HID reports, see the PurePath Wireless Configurator's Human Interface Device panel. See also the help system for available options, and guidelines for creating custom-defined HID reports.

3.7 System Functionality

3.7.1 Power Management

The CC85xx features power-on-reset and brown-out detection circuitry, which generates chip reset at power-up and holds the chip in reset when the power supply drops below a given threshold.

When properly powered but in its lowest power state, OFF, the CC85xx can only wake up from certain external events:

- RESETn pin taken low
- CSn pin taken low
- Exit from power-on-reset or brown-out detection

In the higher power states, manual and automatic power control allows CC85xx and external peripherals to reduce power consumption and/or power down to the OFF state:

- External triggers can include such as push-buttons, remote control and USB SUSPEND
- CC85xx-generated triggers can include such as detection of silent audio inputs/outputs, network inactivity and battery voltage monitoring
- Disabling the power supply to CC85xx and its external peripherals, for instance using a switch in series with the power supply (e.g. battery). Note that when using this option, CC85xx cannot perform a clean shut-down of the external audio device, and also cannot store volume settings and dynamic channel selection to non-volatile memory.

The external audio device can be limited to its lower states for power reduction. If fully disabled by taking it to the OFF audio device state, audio processing and audio clock generation will also be disabled, and radio packets will be significantly shorter.

Below is an overview of the available power states and their effect on audio device and audio processing. Network maintenance is not affected by transitioning between the higher power states (i.e. all but OFF).

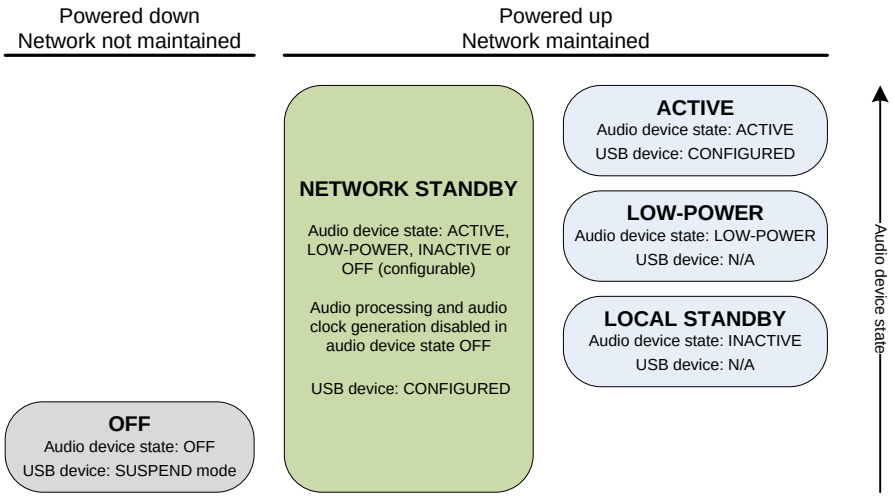


Figure 51 - Power state overview

3.7.1.1 Network Standby Mode

When the protocol master enters the NETWORK STANDBY state, all connected protocol slaves will follow. Network standby mode can be controlled directly at the protocol master and/or through the protocol master

using remote control commands. For example, in a system with a stereo input base station and two wireless mono speakers, a “Standby” push-button on one of the speakers can take all three units into and out of the NETWORK STANDBY state.

The protocol master’s network standby signaling is part of the continuously transmitted network information. While in the NETWORK STANDBY state, the protocol slave’s audio device state is unaffected by temporary or long drop-outs.

Note that network standby mode requires FW 1.3 or later.

3.7.1.2 Battery Voltage Monitoring

Battery voltage monitoring will, when enabled, measure the battery voltage four times per second.

Low battery can be indicated on the network status LED(s) as described in section 3.3.2.1. Hysteresis and delay are used to prevent unstable and glitching low-battery indication. The low battery LED warning is:

- Enabled when the measured battery voltage is below a configurable threshold 4 times in row
- Disabled when the measured battery voltage is above another configurable threshold 20 times in row

Handling of critically low battery voltage depends on operation mode, and is described in the sections below.

3.7.1.3 Autonomous Operation

The following options are available in autonomous operation:

- Button-based control:
    - A power toggle button connected to the CSn pin allows the end-user to take the device into and out of the OFF power state:
        - When the power toggle button is pushed in the OFF power state, the CC85xx will wake up and check whether the button push passes the criteria for a CLICK or HOLD button event (see Table 15). If successful, the CC85xx will enter the ACTIVE power state, and begin network establishment, audio streaming and other related activities. If unsuccessful, the CC85xx will return to the OFF power state as soon as the power toggle button has been released.
        - When the power toggle button is pushed in the ACTIVE power state, and the button push passes the CLICK or HOLD button event criteria, the system will power down. The CC85xx will then put the external audio device in its OFF state, disable the network status LED(s), safely shut down all internal peripherals, and can optionally store the current volume settings to non-volatile memory. It will then enter the OFF power state, where the crystal oscillator and the internal digital power supply are turned off.
        - If the power toggle button is not enabled in the device configuration, the CC85xx will automatically enter the ACTIVE power state at power up, and can only be shut down by disabling the power supply. In this case, volume setting(s) and dynamic channel selection can not be stored to non-volatile memory, and the configured default values will be used at every power-up.
    - A network standby toggle button allows the end-user to switch the entire network into and out of the NETWORK STANDBY power state:
        - On the protocol master, the push-button automatically affects the local power state, which in turn enables or disables network standby signaling to the protocol slaves
        - On protocol slaves, the push-button generates remote control commands to the protocol master, where they have the same effect as a local push-button
        - Handling this state is mandatory, but the enforced audio device state is configurable
- Timeout-based control:
    - If local audio processing finds that the local audio outputs are digitally silent (i.e. all samples are zero, consumers only):

        - The device can enter the LOW-POWER power state after a configurable timeout. This limits the external audio device to the LOW-POWER audio device state.
        - The device can enter the OFF power state after another configurable timeout
    - If local audio processing finds that the local audio inputs are silent (analog or digital silence, producers only):
        - The device can stop transmitting audio slices to reduce current consumption and allow audio consumers to optionally detect this as digital silence and perform further power-reduction after own configurable timeout
        - The device can enter the OFF power state after another configurable timeout
        - Analog silence is defined by maximum peak and maximum average thresholds specified at configuration time
    - If a protocol master is not connected to any protocol slaves
        - The device can enter the LOCAL STANDBY power state after a configurable timeout. This limits the external audio device to the INACTIVE audio device state.
    - If a protocol slave is not connected to a protocol master
        - The device can enter the LOCAL STANDBY power state after a configurable timeout. This limits the external audio device to the INACTIVE audio device state.
        - The device can enter the OFF power state after another configurable timeout
- Critically low battery voltage, for instance below safe operating conditions for the external audio device:
    - If the first battery voltage measurement is below a configurable threshold, power-up will be aborted and the device goes automatically back to the OFF power state
    - If a later battery voltage measurement is below another configurable threshold, the device will automatically enter the OFF power state

3.7.1.4 Host-Controlled Operation

In host-controlled operation, the external host controller is responsible for powering up the CC85xx and controlling the power state. The following EHIF commands are relevant for power control:

- Protocol master and slave: PM_SET_STATE and PM_GET_DATA

The CC85xx will transition from the OFF power state to ACTIVE when CSn or RESETn are pulled low. Using the RESETn pin to enter the ACTIVE state is recommended, as it ensures that the CC85xx is in a defined state (pulling CSn low will only reset the CC85xx when it causes a wake-up from the OFF power state). Notice the boot behavior described in appendix B.1 if using the CSn pin to exit the OFF state.

While in any of the active power states (i.e. all but OFF), the host processor uses

- The EHIF status word to get the current local power state. On protocol slaves, the EVT_PS_CHG event can indicate forced transitions into and out of the NETWORK STANDBY state.
- The PM_GET_DATA command to get information about
    - Latest measured battery voltage
    - The time the local audio outputs have been continuously digitally silent (i.e. all samples are zero)
    - The time the local audio inputs have been continuously silent (analog or digital silence)
        - Analog silence is defined by maximum peak and maximum average thresholds specified at configuration time
    - The time the device has been continuously disconnected from the network or maintained an empty network

- The PM_SET_STATE command to control the power state. No power state changes happen spontaneously except that protocol slaves can be forced into NETWORK STANDBY by the protocol master.

An optional power-down indication signal can tell the host-processor whether the CC85xx is actually powered down. This makes it possible to detect and handle unexpected CC85xx power-ups due to partial system brown-out or electrical noise on the CSn/RESETn pins. The state cannot be obtained otherwise using the EHIF command set, since using any EHIF command will power up the CC85xx.

3.7.1.5 USB Device

USB device power management is controlled by the USB host.

Network standby mode does not affect local audio processing on the USB device, but the associated remote controls and broadcasted standby signal work as in autonomous operation.

When USB enters SUSPEND mode, the protocol master enters the OFF power state with retention of internal states.

The protocol master returns to the previous active state (i.e. ACTIVE or NETWORK STANDBY) when USB RESUME signaling is received on the bus.

Remote wake up is not supported since this would require the network to be active in SUSPEND mode.

A Abbreviations

ADC	Analog to digital converter
BCLK	Bit clock (serial audio interface signal)
CPU	Central processing unit
DAC	Digital to analog converter
DNR	Dynamic range
DSP	Digital signal processor
GIO	General purpose input/output
EHIF	External host interface
FIFO	First in first out
FW	Firmware
HID	Human interface device (USB device class)
I^2C	Inter-IC
I^2S	Inter-IC sound
ID	Identification
LED	Light emitting diode
LJF	Left justified format
LNA	Low noise amplifier
LSB	Least significant bit
MCU	Microcontroller Unit
MCLK	Master clock (serial audio interface signal)
MISO	Master input slave output (SPI interface signal)
MOSI	Master output slave input (SPI interface signal)
MSB	Most significant bit
NVS	Non-Volatile storage
PA	Power amplifier
PCM16	Pulse-code modulation 16-bit
PCME24	Pulse-code modulation 24-bit companded format (15-bit + shift value)
PCMLF	Pulse-code modulation for low frequency
PER	Packet error rate
PPW	PurePath Wireless
PM	Protocol master
PS	Protocol slave
RF	Radio frequency
RJF	Right justified format
SCLK	Serial clock (SPI interface signal)
SLAC	Slightly lossy compression algorithm
S/PDIF	Sony/Philips digital interconnect format

SPI	Serial peripheral interface
SS	Slave select (SPI interface signal)
SW	Software
USB	Universal serial bus
VBR	Variable bit-rate
WASP	Wireless audio streaming protocol
WCLK	Word clock (serial audio interface signal)

B EHIF Reference

The External Host Interface (EHIF) is a set of SPI commands and an interrupt line. The interface lets an external MCU control the CC85xx to a certain extent depending on operational mode. This section describes these EHIF commands in detail.

B.1 Basic SPI Operations

The SPI slave interface is active whenever the CSn pin is asserted. Asserting CSn will also wake up CC85xx from low-power modes and will prevent it from entering low-power modes.

When CSn is deasserted, the MISO pin has pull-up.

The interface operates on whole numbers of bytes, MSB first, and

- Samples serial data on MOSI on the positive edge of SCLK
- Updates serial data on MISO on the negative edge of SCLK

All basic SPI operations except **`SYS_RESET()`**/**`BOOT_RESET()`** and **`GET_STATUS()`** begin with a wait state. For these operations the host controller must, after asserting CSn, wait for MISO to go high (or deassert CSn to abort the operation):

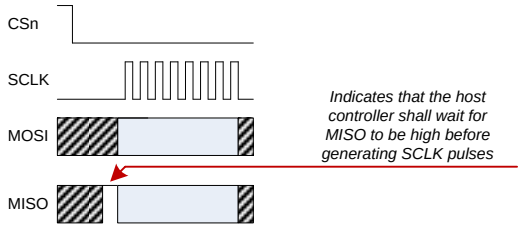


Figure 52 – SPI interface wait state

The duration of the wait state depends on the previous SPI operation. The expected wait state duration is found in the descriptions of the flash programming and external host interface commands.

The following primitives are used as building blocks for flash programming- and external host interface commands.

All fields are most significant byte and bit first (big-endian). Bit field indexes are described as

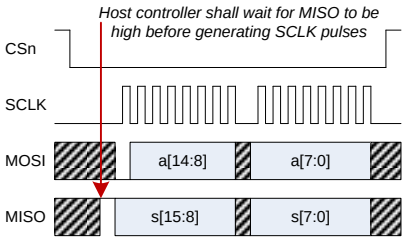
[byte index/range][bit index/range within the byte (array)]

The meaning of the 16-bit status word depends on which CC85xx mode is active; flash programming or external host interface. Refer to these sections for further details.

SET_ADDR(IN: a[15]; OUT: s[16])

Sets the memory address to be used by a subsequent **`WRITE`** or **`READ`** operation.

Note: This operation is only available during flash programming. It is disabled otherwise.



Inputs:

[0:1][15] = 0b0

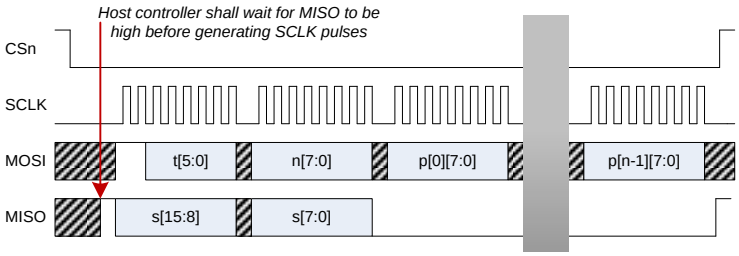
[0:1][14:0] = a: 15-bit memory byte address

Outputs:

[0:1] = s: 16-bit status word

CMD_REQ(IN: t[6], n[8], p[n][8]; OUT: s[16])

Executes a command implemented by the CC85xx bootloader or external host interface. The MISO pin will remain low until the command execution is completed.



Inputs:

[0:1][15:14] = 0b11

[0:1][13:8] = t: Command type

[0:1][7:0] = n: Number of bytes in the parameter field (0-255)

[2:2+(n-1)][7:0] = p: Command parameter field (n bytes)

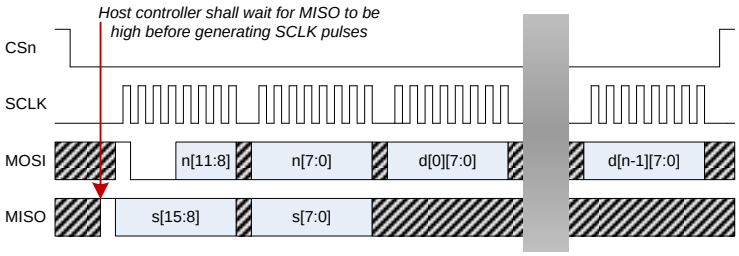
Outputs:

[0:1] = s: 16-bit status word

Other bytes = Don’t care

WRITE(IN: n[12], d[n][8]; OUT: s[16])

Writes the specified number of bytes to the memory location specified by a **SET_ADDR()** operation or the memory buffer setup internally by a **CMD_REQ()** operation.



Inputs:

[0:1][15:12] = 0b1000

[0:1][11:0] = n: Number of bytes in the data field (0-4095)

[2:2+(n-1)][7:0] = d: Data field (n bytes)

Outputs:

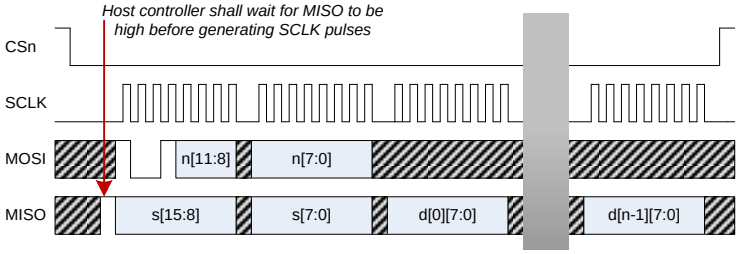
[0:1] = s: 16-bit status word

Other bytes = Don't care

READ(IN: n[12]; OUT: s[16], d[n][8])

Reads the specified number of bytes from the result of a **CMD_REQ** operation.

The number of bytes in the data field must be known prior to executing the command. The host controller can choose to set the "n" in parameter lower than the expected number of bytes if it is only interested in reading the first part of the result.



Inputs:

[0:1][15:12] = 0b1001

[0:1][11:0] = n: Number of bytes in the data field (0-4095)

Other bytes = Don't care

Outputs:

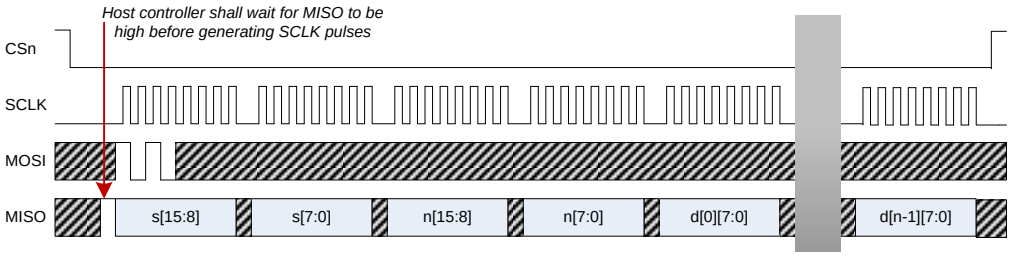
[0:1] = s: 16-bit status word

[2:2+(n-1)] = d: Data field (n bytes)

READBC(OUT: s[16], n[16], d[n][8])

Reads an unspecified number of bytes from the result of a **CMD_REQ** operation.

The number of bytes in the data field is output during the operation. The host controller can choose to abort the operation before “n” bytes have been read out if it is only interested in reading the first part of the result.



Inputs:

[0:1][15:12] = 0b1010

Other bits = Don’t care

Outputs:

[0:1] = s: 16-bit status word

[2:3] = n: Number of bytes in the data field (0-4095)

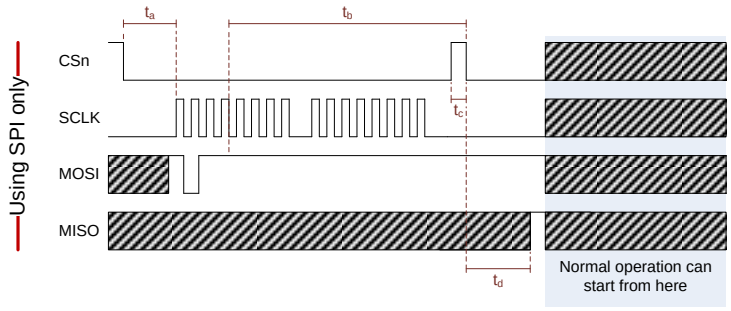
[4:4+(n-1)] = d: Data field (n bytes)

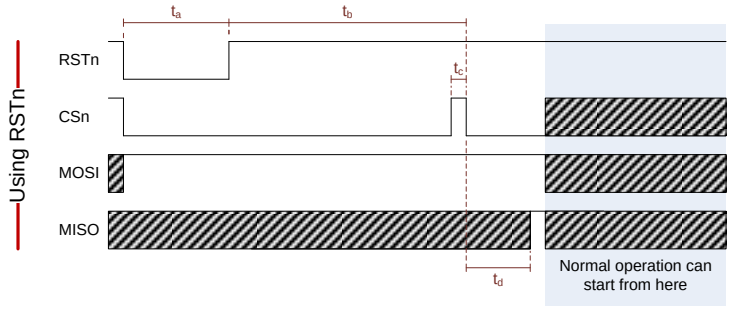
SYS_RESET()

Resets CC85xx and starts the audio streaming application where the external host interface is available.

If the SPI/external host interface is unused, MOSI should be tied high to prevent accidental BOOT_RESET(). SYS_RESET() can then be performed by simply applying a low pulse on RSTn.

While CC85xx is powered down, a falling edge on CSn will have the same effect as asserting RSTn. This is used by the power toggle button (on CSn) to power up the CC85xx.





Inputs (“Using SPI only”):

[0:1] = 0b1011.1111.1111.1111

Outputs (“Using SPI only”):

[0:1] = Don’t care

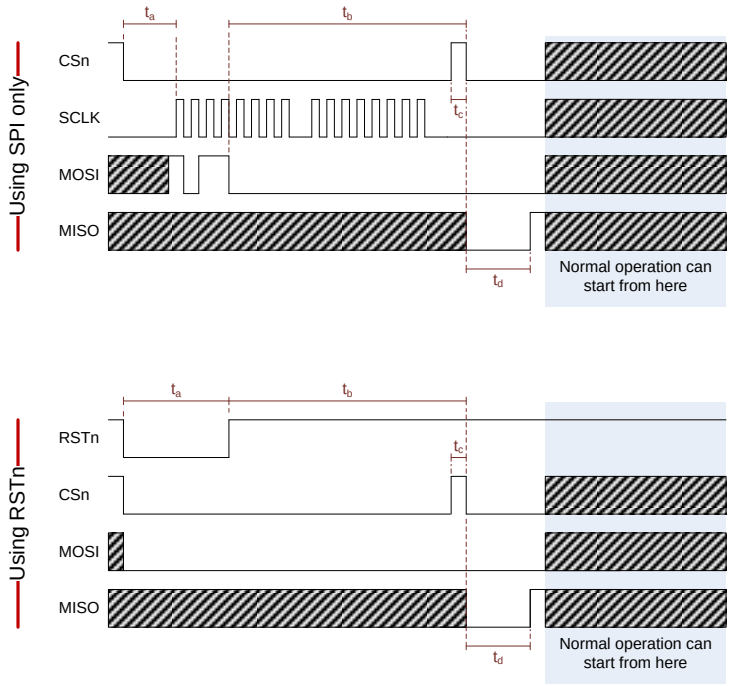
When performing SPI reset, the internal reset sequence begins shortly after the first nibble of the first byte. Note, however, that *the entire SPI sequence must be completed* to generate a safe reset cycle.

The MOSI signal must remain high from the start of t_b until MISO is high at the end of t_d.
Note the following timing requirements:

- t_a: 1 us or more. If CC85xx is powered down, add a 2 ms delay before proceeding
- t_b: 4 us or more
- t_c: 1 us or more
- t_d: Wait until MISO is high before proceeding. For small values of t_b, MISO will be low at the start of t_d, while for large values of t_b, MISO will be high already at the start of t_d.

BOOT_RESET()

Resets CC85xx and enters the bootloader where the flash programming interface is available.



Inputs (“Using SPI only”):

[0:1] = 0b1011.0000.0000.0000

Outputs (“Using SPI only”):

[0:1] = Don’t care

When performing SPI reset, the actual reset sequence begins shortly after the first nibble of the first byte. Note, however, that *the entire SPI sequence must be completed* to generate a safe reset cycle.

The MOSI signal must remain low from the start of t_b until MISO goes high at the end of t_d.

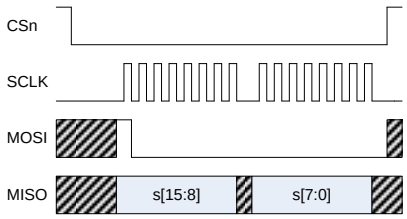
Note the following timing requirements:

- t_a: 1 us or more. If CC85xx is powered down, add a 2 ms delay before proceeding
- t_b: Between 4 us and 10 us
- t_c: Approx. 1 us
- t_d: Wait until MISO goes high before proceeding.
  MISO should always be low at the start of t_d.

GET_STATUS(OUT: s[16])

Returns the SPI status word.

Depending on the situation, the wait state can be used to detect that a command has been completed, or it can be ignored when monitoring the SPI status word during command execution.



Inputs:

[0:1] = 0b1000.0000.0000.0000

Outputs:

[0:1] = s: 16-bit status word

B.2 Boot Behavior and Flash Programming

This section describes boot behavior and flash programming, which is relevant for implementing

- Production programmers
- In-application programming by external host controller, e.g. over-the-air download using flash image received through the data side-channel. This allows for instance firmware upgrades on all speakers distributed from a base station.
    - Note that CC85xx only provides the flash programming interface and a general datagram transfer mechanism. The host controllers are responsible for implementing a communications protocol which ensures that:
        - Only compatible host controllers communicate with each other
        - The correct application image is programmed
        - The programmed application image is complete and without errors. Programming a corrupted image will cause CC85xx to malfunction.
    - The host controller must also ensure that the power supply is good and stable

B.2.1 On-Chip Memory

CC85xx stores the application image generated by the PurePath Wireless Configurator in on-chip flash memory. Flash programming is supported through an on-chip bootloader stored in ROM.

Figure 53 shows the memory areas that are relevant for boot behavior and flash programming. Note that this figure is a simplification of the actual memory layout, and is only valid for the operations described in section B.2.3. Do not attempt to access other address ranges as this will cause random behavior and can potentially damage the device physically.

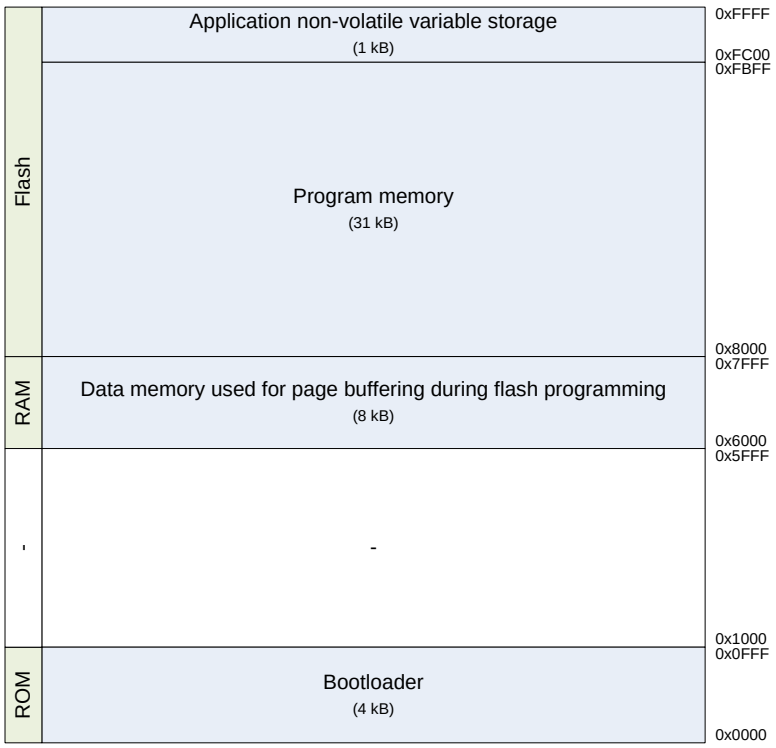


Figure 53 - Memory layout

B.2.2 Boot Behavior

A command is built up of basic SPI operations (CMD_REQ, READ, READBC, and WRITE). These operations are described in section B.1.

Reset Event

The boot sequence will usually start as the result of power-up, waking up from the OFF power state or a manual reset using SYS_RESET() or BOOT_RESET() (see section B.1 for further details).

A reset event can occur “out of the blue” during active operation, as the result of a brown-out condition. Depending on the activity on the SPI interface at that time, the result can either be equivalent to SYS_RESET() or BOOT_RESET().

To prevent host controllers from causing damage to CC85xx by unknowingly entering the bootloader and executing random commands (potentially resulting in irreversible flash corruption), CC85xx implements a one-shot bootloader unlocking mechanism to ensure that the bootloader cannot be entered unintentionally. This unlocking procedure is described below.

Boot Sequence

The flow diagram below describes the boot sequence and the possible outcomes:

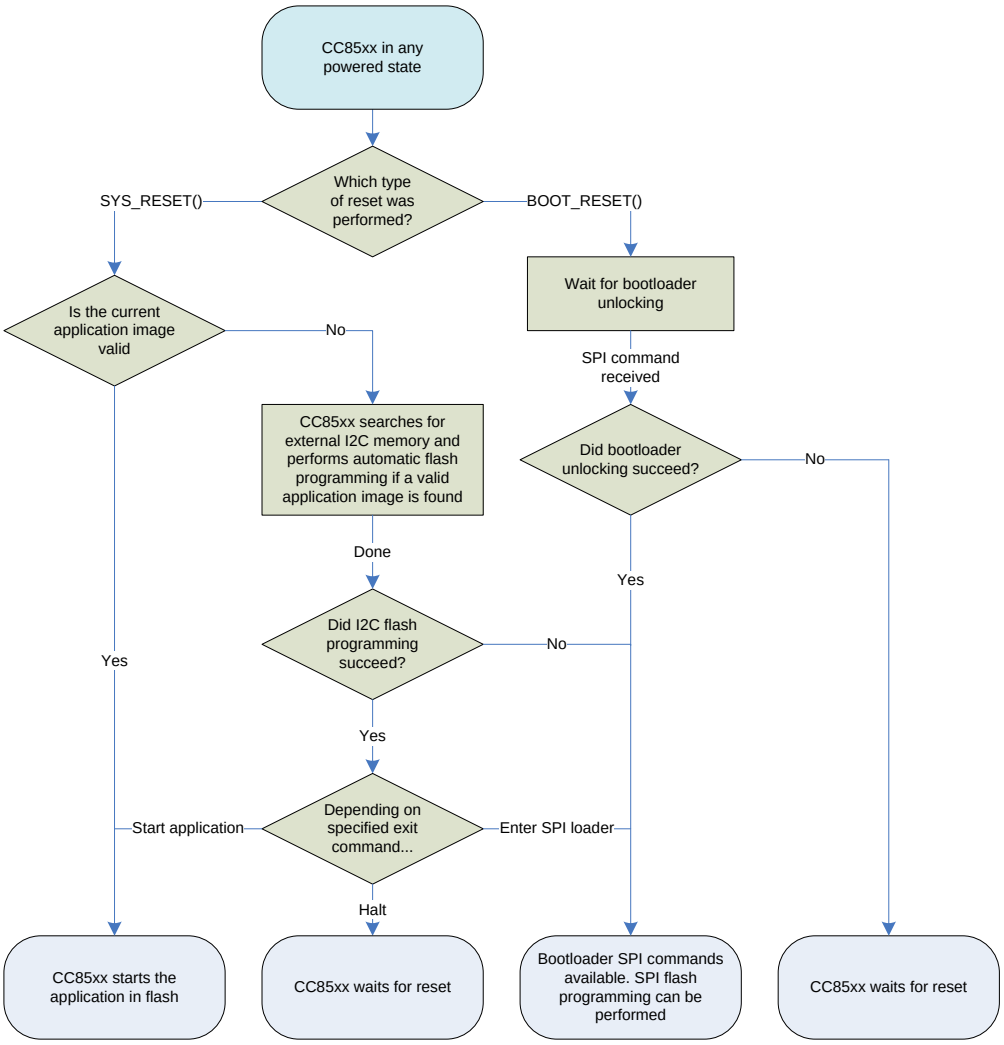


Figure 54 - Boot sequence

Bootloader Unlocking

Unless there is no valid flash image already loaded, the host controller must run BL_UNLOCK_SPI() to enter the bootloader and get access to the commands for SPI flash programming.

Command	**BL_UNLOCK_SPI**
Description	Unlocks the SPI commands provided by the bootloader, which allows for flash programming through the SPI slave interface. The command is one-shot, which means that the chip freezes if the command fails and must be reset to make another attempt. Relevant status word (sw) values are: • 0x8021: Waiting for the BL_UNLOCK_SPI() command • 0x8020: Unlocking succeeded; commands described in section B.2.3 are accepted. • 0x0022: Unlocking failed

SPI Operations
 	
(pars)
	`CMD_REQ(0x00, 4, pars, sw)`
	
	[Byte][bit] index | **Field** | **Description**
	`[0:3]` | `KEY[31:0]` | `Fixed value: 0x2505B007 (magic number)`

Execution time	Normally < 1 μs
Usage limitations	Command will remain unchanged across all future firmware revisions
Notes	This command is only available and only relevant after a BOOT_RESET()

B.2.3 Flash Programming Through SPI Slave Interface

Flash Programming through SPI Slave Interface

Flash programming is performed using the commands described below and in addition some routines based on the basic SPI operations defined in section B.1.

The DI_GET_CHIP_INFO EHIF command (see section B.3.1) can be used while the bootloader is active to extract information about the CC85xx device and the currently loaded firmware image.

Command	**BL_FLASH_MASS_ERASE**
Description	Erases all flash contents including program memory and application non-volatile storage. This command shall always be executed prior to executing the BL_FLASH_PAGE_PROG command loop to program the flash. If not the flash contents will be incorrect and it can also affect the number of erase cycles supported by the flash memory. Relevant status word (sw) values are: • 0x0002: Mass erase in progress • 0x8003: Mass erase completed successfully • 0x8004: Command failed due to incorrect KEY parameter

SPI Operations
 	
(pars)
	`CMD_REQ(0x03, 4, pars, sw)`
	
	[Byte][bit] index | **Field** | **Description**
	`[0:3]` | `KEY[31:0]` | `Fixed value: 0x25051337 (magic number)`

Execution time	< 25 ms
Usage limitations	Command will remain unchanged across all future firmware revisions
Notes	This command is available only in the bootloader after unlocking has been performed

Command	BL_FLASH_PAGE_PROG
Description	Programs a single 1 kB flash page using the data which has been written to the given RAM location. The command is repeated for each flash page with increasing flash address and new data in RAM. Relevant status word (sw) values are: • 0x000A: Programming in progress • 0x800B: Programming completed successfully • 0x800C: Command failed due to incorrect KEY parameter

SPI Operations
 	
(pars)
	`CMD_REQ(0x07, 10, pars, sw)`
	
	[Byte][bit] index / **Field** / **Description**
	`[0:1]` / `RAM_ADDR[15:0]` / `Location in RAM where the data to be programmed is stored (0x6000-0x7C00). Note that the address must be modulo 4 (i.e. the two LSBs must be 0).`
	`[2:3]` / `FLASH_ADDR[15:0]` / `Address of start of flash page to be programmed: 0x8000 + (n x 0x0400), where n is between 0 and 30`
	`[4:5]` / `DWORD_COUNT[15:0]` / `Fixed value: 0x0100 (1 kB page size)`
	`[6:9]` / `KEY[31:0]` / `Fixed value: 0x25051337 (magic number)`

Execution time	< 10 ms
Usage limitations	Command will remain unchanged across all future firmware revisions
Notes	This command is available only in the bootloader after unlocking has been performed, and should only be performed on flash pages not programmed after the last successful BL_FLASH_MASS_ERASE command.

Command	BL_FLASH_VERIFY
Description	Verifies that the flash programming succeeded by calculating a CRC32 checksum over the entire image, and comparing it against the expected checksum value, which is located at the end of the image. The command requires the size of the unpadded FW image, *nImageBytes*, which is found in the Intel HEX file at address 0x801C as a 32-bit value. The FW Configurator tool generates Intel HEX files with a CRC32 checksum added after the last byte of FW image and then pads the rest of the memory space with 0xFF bytes. Relevant status word (sw) values are: • 0x000D: Verification in progress • 0x800E: Verification completed successfully (the image is intact) • 0x800F: Command failed due to checksum mismatch

SPI Operations
 	
(pars)
	`CMD_REQ(0x0F, 8, pars, sw)`
	
	[Byte][bit] index / **Field** / **Description**
	`[0:3]` / `DATA_ADDR[31:0]` / `Fixed value: 0x00008000`
	`[4:7]` / `BYTE_COUNT[31:0]` / `Number of bytes to calculate the checksum over = nImageBytes (32-bit value found in the Intel HEX file at address 0x801C)`
	
	If desired, the CRC32 value calculated over BYTE_COUNT bytes from address DATA_ADDR can be read out with an optional subsequent read phase:
	
	`READ(4, sw, data)`
	
	[Byte][bit] index / **Field** / **Description**
	`[0:3]` / `CRC_VAL` / Big-endian byte order CRC32 value calculated over BYTE_COUNT bytes from address DATA_ADDR.

Execution time	< 50 ms
Usage limitations	Command will remain unchanged across all future firmware revisions
Notes	This command is available only in the bootloader after unlocking has been performed.

Flash Programming Algorithm

If the programming is based on an Intel HEX file output from the PurePath Wireless Configurator, the image starts at address 0x8000 and ends at address 0xFBFF. Ignore any other data found in the HEX file.

The procedure starts at the lowest flash address and ends with the highest flash address.

Below is the complete flash programming algorithm, described in a C-like syntax. Parameters described as fixed in the command set defined above are not included in the parameter lists. The SPI status word is not checked in this algorithm, but should be in a real implementation.

```
// Enter the bootloader
BOOT_RESET();
BL_UNLOCK_SPI();

// Erase the flash
BL_FLASH_MASS_ERASE();

// For each flash page ...
for (int n = 0; n < 31; n++) {

    // Write page data to the start of the available RAM area
    SET_ADDR(0x6000, sw);
    WRITE(0x400, data to be programmed into flash page n, sw);

    // Program the page
    BL_FLASH_PROG_PAGE(RAM_ADDR = 0x6000, FLASH_ADDR = 0x8000+(n*0x400))
}

// Verify the flash contents
BL_FLASH_VERIFY(BYTE_COUNT = value read from the HEX file);

// Done, perform SYS_RESET() to start the application
```

B.2.4 Flash Programming Through I²C EEPROM Memory

Given that no valid image has been loaded, the bootloader also allows CC85xx to program itself from data stored in an I²C memory device.

While programming through the SPI slave interface is recommended due to better security, flexibility and speed, programming through the I²C interface can still be used as an alternative way to perform production programming. Security observability can be improved significantly by monitoring the programmed devices using the GET_STATUS() SPI operation.

The programming starts after power-up or SYS_RESET().I²C programming is usually one-shot, because the quick validity check after SYS_RESET() in Figure 54 does not detect partial programming. Any new attempts will require a BL_FLASH_MASS_ERASE() operation to be performed.

I²C Memory Device Requirements

The I²C clock speed is 400 kHz. The I²C memory device is assumed to be a Smart Serial I²C device, with a 16-bit memory address, and an I²C slave address + R/W byte formatted as follows:

[1010 | AAA | R/W bit], where AAA is between 0 and 7

The programming routine does not implement proper handling of unexpected events on the I²C bus. This means that other I²C masters must not interfere while I²C programming is running. The I²C programming will release the bus by generating a STOP condition between read operations. It is assumed that I²C communication errors will result in a failing CRC32 check when programming is completed.

CC85xx will search for an I²C memory device, starting at the first address. If multiple devices are present on the bus, only the one with the lowest address will be detected.

If the search fails, the bootloader will enter SPI mode and allow the commands described above to be used.

I²C Memory Image Formatting

The data contained in the I²C memory shall start at address 0x0000, and be formatted as follows:

Table 24 - I²C Memory Format

Field	Address	Size (bytes)	Description
DWC	0x0000	2 B	Total number of 32-bit words in the flash image to be programmed, including the CRC32 checksum.
CMD	0x0002	2 B	Exit command: 0x0000 – Wait for reset 0x0001 – Start application 0x0002 – Stay in the bootloader and accept the SPI commands described above.
DATA+CRC	0x0004	DWC * 4	Data taken from the Intel HEX file generated by the PurePath Wireless Configurator, starting at address 0x8000 and including the CRC32 checksum at the end of the image.

The 16-bit DWC and CMD fields are big-endian.

A validity check is performed before flash mass erase begins:

- The DWC and CMD fields must be sane
- Some constant values that are included in all CC85xx flash images must match the expected values

If the validity check fails, the bootloader will enter SPI mode and remain there.

If the validity check succeeds, the CC85xx flash is erased and then programmed page by page.

After programming, a CRC32 check of the programmed image will be performed. If the check fails, the bootloader will freeze and wait for reset. The SPI status at that time will indicate the failure.

Relevant SPI Status Word Values

The SPI status word is updated during I²C programming. The value of MSB of the status word depends on the exit command can be either 0 or 1 depending on the exit command, and can safely be ignored:

- 0x0005: I²C programming in progress
- 0x0006: I²C programming completed successfully (not observable if the exit command starts the application after a reset)
- 0x0007: Verification after programming failed. Mass erase must be performed before another attempt can be made.
- 0x0008: No I²C memory device was found
- 0x0009: I²C memory device was found, but the image is invalid

B.3 EHIF Command Set

A command is built up of basic SPI operations (CMD_REQ, READ, READBC, and WRITE). These operations are described in section B.1.

A status word is clocked out on the MISO pin every time an SPI operation is clocked in on the MOSI pin (see Table 25 for flags and fields in the status word). The most significant flag in this status word (CMDREQ_RDY) is automatically output on the MISO pin whenever CSN is pulled low. This flag (pin) must be high before any operation is clocked in. The flags in the status word with `EVT_` prefix are event flags that can be configured (by the EHC_EVT_MASK command) to contribute to the EHIF interrupt line.

Refer to section 3.2 for more details about the EHIF status word.

Table 25 - EHIF SPI status word

Bit(s)	Name	Must clear flag	Description
15	CMDREQ_RDY	-	Indicates whether EHIF is ready for new SPI operation (CMD_REQ, WRITE, READ or READBC.
14:12	-	-	Reserved, ignore value.
11:9	PWR_STATE[2:0]	-	Current device power state: 5: ACTIVE – Device is fully active 4: LOW-POWER – Audio device is in LOW-POWER state 3: LOCAL STANDBY – Audio device is in INACTIVE state 2: NETWORK STANDBY – Entire network is in standby mode 0: OFF – Device is powered down, requiring external wake-up Other values (1, 6 and 7) are reserved.
8	WASP_CONN	N	PM: Set if audio network contains at least one slave. PS: Set if connected to an audio network.
7	EVT_DSC_RX_AVAIL	N	Set whenever the data side-channel's reception queue contains one or more datagrams (ready to be read out.
6	EVT_DSC_TX_AVAIL	N	Set whenever there is space for one or more new datagrams in the data side-channel's transmission queue.
5	EVT_DSC_RESET	Y	Data side-channel was reset
4	EVT_SPI_ERROR	Y	Indicates invalid SPI command/parameters or unexpected read/write operations
3	EVT_VOL_CHG	Y	Volume for device has changed
2	EVT_PS_CHG	Y	Power state of device changed (current power state available in PWR_STATE).
1	EVT_NWK_CHG	Y	PM: A slave has joined or dropped out of the network. PS: Join attempt failed or disconnected after successful join operation.
0	EVT_SR_CHG	Y	Audio sample rate has changed.

Different subsets of SPI command are available in the different modes of operation (autonomous, host-controlled and production test), but the interface is typically only used in host-controlled operation and production test.

Table 26 - EHIF command summary

Command	A[3]	H	P	Basic SPI operations	Description
Device information					
DI_GET_CHIP_INFO	✓	✓	✓	CMD_REQ(0x1F, 2, pars, sw) READ(24, sw, data)	Returns hardware/firmware information
DI_GET_DEVICE_INFO	✓	✓	✓	CMD_REQ(0x1E, 0, sw) READ(12, sw, data)	Returns unique device ID and manufacturer-specific information
EHIF control					
EHC_EVT_MASK	✓	✓	✓	CMD_REQ(0x1A, 2, pars, sw)	Configures EHIF interrupt pin event mask
EHC_EVT_CLR	✓	✓	✓	CMD_REQ(0x19, 1, pars, sw)	Clears EHIF event flags
Audio network control/status					
NWM_DO_SCAN	✗	✓	✗	CMD_REQ(0x08, 16, pars, sw) READBC(sw, nBytes, data)	Performs a scan of the whole 2.4 GHz band searching for audio networks within range and returns a list of those found
NWM_DO_JOIN	✗	✓	✗	CMD_REQ(0x09, 18, pars, sw)	Joins a specific audio network or the first available one meeting specified criteria. Also used to leave a network
NWM_GET_STATUS	✗	✓	✗	CMD_REQ(0x0A, 0, sw) READBC(sw, nBytes, data)	Returns information about current audio network status and information about other nodes in network
NWM_ACH_SET_USAGE	✗	✓	✗	CMD_REQ(0x0B, 16, pars, sw)	For protocol slaves: Determine which audio channels in network to produce/consume and maps them to serial audio interface channels
NWM_CONTROL_ENABLE	✗	✓	✗	CMD_REQ(0x0C, 2, pars, sw)	For protocol masters: Enables/disables formation/maintenance of the audio network
NWM_CONTROL_SIGNAL	✗	✓	✗	CMD_REQ(0x0D, 2, pars, sw)	For protocol masters: Enables/disables pairing signaling
NWM_SET_RF_CH_MASK	✓	✓	✗	CMD_REQ(0x0E, 4, pars, sw)	For protocol masters: Modifies currently used or to be used RF channel mask (with 6-18 channels) or disables RF (when setting 0-5 channels)
Remote control					
RC_SET_DATA	✗	✓	✗	CMD_REQ(0x2D, 13, pars, sw)	For protocol slaves: Communicates remote control button/keyboard states/mouse position+buttons to the protocol master
RC_GET_DATA	✗	✓	✗	CMD_REQ(0x2E, 1, pars, sw) READ(13, sw, data)	For protocol masters: Retrieves the remote control button/keyboard states/mouse position+buttons for a specific protocol slave
Data side-channel					
DSC_TX_DATAGRAM	✗	✓	✗	CMD_REQ(0x04, 5, pars, sw) WRITE(nBytes, data, sw)	Queues data side-channel datagram for transmission if space is available
DSC_RX_DATAGRAM	✗	✓	✗	CMD_REQ(0x05, 0, sw) READBC(sw, nBytes, data)	Gets first datagram in data side-channel reception queue (if available)
Power management					
PM_SET_STATE	✗	✓	✗	CMD_REQ(0x1C, 1, pars, sw)	Sets/limits the current power state
PM_GET_DATA	✗	✓	✗	CMD_REQ(0x1D, 0, sw) READBC(sw, nBytes, data)	Returns power management related information (audio input silence, audio output silence, network idle time and battery voltage)
Volume control					
VC_SET_VOLUME	✗	✓	✗	CMD_REQ(0x17, 4, pars, sw)	Sets global (protocol master only) and local input or output volume
VC_GET_VOLUME	✗	✓	✗	CMD_REQ(0x16, 1, pars, sw) READ(2, sw, data)	Returns global (protocol master only) and local input or output volume
RF/audio performance statistics					
PS_RF_STATS	✓	✓	✗	CMD_REQ(0x10, 0, sw) READBC(sw, nBytes, data)	Outputs network and RF statistics, and resets counters afterwards
PS_AUDIO_STATS	✓	✓	✗	CMD_REQ(0x11, 0, sw) READBC(sw, nBytes, data)	Outputs audio processing and audio streaming performance statistics, and reset counters afterwards
Calibration					
CAL_SET_DATA	✓	✓	✗	CMD_REQ(0x28, 5, pars, sw)	Writes, among other, the TX output power override setting
CAL_GET_DATA	✓	✓	✗	CMD_REQ(0x29, 0, sw) READ(4, data, sw)	Reads, among other, the TX output power override setting
Utilities					

[3] A=Autonomous operation, H=Host-controlled operation, P=Production test

Command	A[3]	H	P	Basic SPI operations	Description
IO_GET_PIN_VAL	✗	✓	✗	CMD_REQ(0x2A, 0, sw)	Reads the current value of IO pins
NVS_GET_DATA	✓	✓	✗	CMD_REQ(0x2B, 1, pars, sw) READ(4, sw, data)	Reads data from non-volatile storage
NVS_SET_DATA	✓	✓	✗	CMD_REQ(0x2C, 5, pars, sw)	Writes data to non-volatile storage
RF test					
RFT_TXTST_CW	✗	✗	✓	CMD_REQ(0x15, 3, pars, sw)	Enable/disable infinite RF continuous wave generation at a specific RF frequency with a specific output power.
RFT_TXTST_PN	✗	✗	✓	CMD_REQ(0x14, 3, pars, sw)	Enable/disable infinite RF pseudorandom modulated data generation at a specific RF frequency with a specific output power
RFT_RXTST_CONT	✗	✗	✓	CMD_REQ(0x25, 2, pars, sw)	Enables continuous reception for regulatory compliance testing
RFT_RXTST_RSSI	✗	✗	✓	CMD_REQ(0x26, 1, pars, sw) READ(1, sw, data)	Measures the RSSI at a specific frequency
RFT_NWKSIM	✗	✗	✓	CMD_REQ(0x27, 14, pars, sw)	Enable/disable infinite test of single-node network simulator. Used for regulatory compliance testing.
RFT_TXPER	✗	✗	✓	CMD_REQ(0x13, 8, pars, sw)	Runs the transmitter side of the packet error rate test. Used to evaluate radio performance
RFT_RXPER	✗	✗	✓	CMD_REQ(0x12, 11, pars, sw) READ(244, sw, data)	Runs the receiver side of the packet error rate test. Used to evaluate radio performance
Audio test					
AT_GEN_TONE	✗	✗	✓	CMD_REQ(0x20, 5, pars, sw)	Generates an output tone with specified frequency and amplitude at the specified logical audio channel.
AT_DET_TONE	✗	✗	✓	CMD_REQ(0x21, 1, pars, sw) READ(4, sw, data)	Measures the frequency and amplitude of a tone on the specified logical audio channel.
IO test					
IOTST_INPUT	✗	✗	✓	CMD_REQ(0x22, 4, pars, sw) READ(4, sw, data)	Returns logical value on selected pins.
IOTST_OUTPUT	✗	✗	✓	CMD_REQ(0x23, 8, pars, sw)	Drives logical value on selected pins

Most EHIF commands are executed by the main CPU with medium/low priority; hence the exact time of execution is non-deterministic. All commands should complete within 10 ms + any command specific timeout or other command specific delays.

B.3.1 Device Information Commands

Command	DI_GET_CHIP_INFO		
Description	Returns hardware/firmware information.		
SPI Operations	`CMD_REQ(0x1F, 2, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0:1]	–	Reserved, must be 0x00B0
	`READ(24, sw, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	[0:1]	FAMILY_ID[15:0]	Hardware family ID. Returns 0x2505
	[2:3][15:8]	–	Reserved, ignore value
	[2:3][7:4]	SIL_MAJ_REV[3:0]	Silicon major/minor revision. First released silicon is revision 2.0
	[2:3][3:0]	SIL_MIN_REV[3:0]	
	[4:7][31:16]	ROM_TYPE[15:0]	ROM image type and major and minor revision. Type is 0xB007 and revision 2.0 for first released silicon
	[4:7][15:8]	ROM_MAJ_REV[7:0]	
	[4:7][7:0]	ROM_MIN_REV[7:0]	
	[8:11][31:16]	FW_TYPE[15:0]	Type and revision of FW stored in flash on device. Type is 0xFFFF for an unprogrammed device and 0x2505 for a programmed device.
	[8:11][15:12]	FW_MAJ_REV[3:0]	
	[8:11][11:8]	FW_MIN_REV[3:0]	
	[8:11][7:0]	FW_PATCH_REV[7:0]	
	[12:15]	–	Reserved, ignore value
	[16:19]	FW_IMAGE_SIZE[31:0]	Size of FW image in bytes (only valid if `FW_TYPE` is valid)
	[20:21]	CHIP_ID[15:0]	ID of silicon device. Returns 0x8520 for first released silicon.
	[22:23]	CHIP_CAPS[15:0]	Chip capabilities. For internal use only.
Related events	-		
Usage limitations	Available even on un-programmed devices Available in all operational modes		
Notes	Command will remain unchanged across all future firmware revisions This command is available even on unprogrammed devices and under all circumstances.		

Command	DI_GET_DEVICE_INFO		
Description	Returns unique device ID and manufacturer-specific information		
SPI Operations	`CMD_REQ(0x1E, 0, sw)`		
	`READ(12, sw, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	[0:3]	DEVICE_ID[31:0]	Unique 32b device ID. Values 0x00000000 or 0xFFFFFFFF can never occur.
	[4:7]	MANF_ID[31:0]	Unique 32b manufacturer ID[4] Value 0xFFFFFFFF means not set/do not care.
	[8:11]	PROD_ID[31:0]	Product/family ID as determined by manufacturer Value 0xFFFFFFFF means not set/do not care.
Related events	-		
Usage limitations	Available in all operational modes		
Notes			

[4] To ensure global uniqueness of Manufacturer ID, the manufacturer should select the unique device ID of any one purchased device.

B.3.2 EHIF Control Commands

Command	EHC_EVT_MASK		
Description	Configures the EHIF interrupt pin event mask. The interrupt pin is active when at least one event flag and its mask bit are both high ((EVENT_FLAGS & EVENT_MASK) != 0). The polarity of the interrupt pin is specified by the EHIF_IRQ_POL field. The EHIF interrupt pin is statically mapped to an available GIO pin or the MISO pin in the PurePath Wireless Configurator when generating firmware image.		
SPI Operations	CMD_REQ(0x1A, 2, pars, sw)		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0][7:1]	-	Reserved, write 0
	[0][0]	EHIF_IRQ_POL	Determines polarity of interrupt pin: 0: Active low 1: Active high
	[1][7]	MSK_DSC_RX_AVAIL	If set, EVT_DSC_RX_AVAIL will contribute to interrupt pin output (level-based, so will auto clear)
	[1][6]	MSK_DSC_TX_AVAIL	If set, EVT_DSC_TX_AVAIL will contribute to interrupt pin output (level-based, so will auto clear)
	[1][5]	MSK_DSC_RESET	If set, EVT_DSC_RESET will contribute to interrupt pin output
	[1][4]	MSK_SPI_ERROR	If set, EVT_SPI_ERROR will contribute to interrupt pin output
	[1][3]	MSK_VOL_CHG	If set, EVT_VOL_CHG will contribute to interrupt pin output
	[1][2]	MSK_PS_CHG	If set, EVT_PS_CHG will contribute to interrupt pin output
	[1][1]	MSK_NWK_CHG	If set, EVT_NWK_CHG will contribute to interrupt pin output
	[1][0]	MSK_SR_CHG	If set, EVT_SR_CHG will contribute to interrupt pin output
Related events	All events		
Usage limitations	Available in all operational modes		
Notes	The mask setting takes effect immediately so executing this command may cause an immediate interrupt.		

Command	EHC_EVT_CLR		
Description	Clear EHIF event flags		
SPI Operations	CMD_REQ(0x19, 1, pars, sw)		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0][7:6]	-	Reserved, write 0
	[0][5]	MSK_DSC_RESET	If set, clears any MSK_DSC_RESET flag
	[0][4]	MSK_SPI_ERROR	If set, clears any MSK_SPI_ERROR flag
	[0][3]	MSK_VOL_CHG	If set, clears any MSK_VOL_CHG flag
	[0][2]	MSK_PS_CHG	If set, clears any MSK_PS_CHG flag
	[0][1]	MSK_NWK_CHG	If set, clears any MSK_NWK_CHG flag
	[0][0]	MSK_SR_CHG	If set, clears any MSK_SR_CHG flag
Related events	All events		
Usage limitations	Available in all operational modes		
Notes	The flag clearing takes effect before the CMDREQ_RDY flag is asserted so host controller interrupt can safely be re-enabled at that point in time.		

B.3.3 Audio Network Control and Status Commands

Command	NWM_DO_SCAN
Description	Used by protocol slaves to perform a scan of the whole 2.4 GHz band for PurePath Wireless networks within range matching filtering criteria, and returns a list of those found. The scan will terminate after SCAN_TO milliseconds or when a maximum of SCAN_MAX networks have been found, whichever comes first. Scanning causes any existing network connection to close down. To perform button-based pairing as in autonomous operation (with pairing signal from protocol master), set SCAN_MAX = 1, REQ_WPM_PAIRING_SIGNAL = 1 and REQ_RSSI = -128. If successful, perform NWM_DO_JOIN with the returned DEVICE_ID. To perform proximity-based pairing as in autonomous operation (without pairing signal from protocol master), set SCAN_MAX = 1, REQ_WPM_PAIRING_SIGNAL = 0 and REQ_RSSI >> -128. If successful, perform NWM_DO_JOIN with the returned DEVICE_ID.

Command	NWM_DO_SCAN		
SPI Operations	`CMD_REQ(0x08, 16, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0:1][15:12]	SCAN_MAX[3:0]	Maximum number of networks to locate before ending scan, SCAN_MAX∈[1,8].
	[0:1][11:0]	SCAN_TO[11:0]	Timeout of scan operation in increments of 10 ms.
	[2:5]	MANF_ID[31:0]	Manufacturer ID filtering criteria. A value of 0 lists all networks regardless of manufacturer ID, any other value lists only networks with a matching manufacturer ID.
	[6:9]	PROD_ID_MASK[31:0]	Product/family ID mask and reference value to be used as filtering criteria. Only networks with PROD_ID&PROD_ID_MASK == PROD_ID_REF&PROD_ID_MASK will be listed. Thus a PROD_ID_MASK value of 0 will disable filtering based on product ID.
	[10:13]	PROD_ID_REF[31:0]	
	[14][7:1]	–	Reserved, write 0.
	[14][0]	REQ_WPM_PAIR_SIGNAL	Require that the protocol master is signaling that it is attempting to pair with a new slave. Set to 0 to disable the criteria.
	[15]	REQ_RSSI	Required RSSI minimum value of the first received master packet when acquiring network during scan operation (signed 2's complement, 0 → ~0 dBm). Set to -128 dBm to disable the criteria.
	`READBC(sw, nBytes, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	For each network found during scan, *i* =[0,nBytes/28). nBytes=0 means no networks were found.		
	[0+28*i*:3+28*i*]	DEVICE_ID[31:0]	Network ID of current network (device ID of master) A value of 0 means no active network connection
	[4+28*i*:7+28*i*]	MANF_ID[31:0]	Manufacturer ID of protocol master (0=not connected)
	[8+28*i*:11+28*i*]	PROD_ID [31:0]	Product ID of protocol master (0= not connected)
	[12+28*i*][7:5]	WPM_PM_STATE	Protocol master's assumed power state (possible values): 5: ACTIVE (1+ logical channel subscriptions active) 4: LOW_POWER (0 logical channel subscriptions active) 3: LOCAL_STANDBY (network currently empty) 2: NWK_STANDBY (network standby signaling active)
	[12+28*i*][4]	WPM_DSC_EN	Indicates that master supports data side channel communication
	[12+28*i*][3]	WPM_MFCT_FILT	Indicates that master only accepts protocol slaves with matching manufacturer ID into network
	[12+28*i*][2]	WPM_PAIR_SIGNAL	Indicates that master is signaling that it is attempting to pair with a new slave
	[12+28*i*][1]	WPM_ALLOWS_JOIN	Indicates that master can accept more protocol slaves into network
	[12+28*i*][0]	–	Reserved, ignore value
	[13+28*i*:14+28*i*]	ACH_SUPPORT[15:0]	Bitmask providing information about which audio channels supported by network. The indexes correspond to the audio channel type defined in Table 9.
	[15+28*i*][7]	ACH_ACTIVE[0]	Flag that indicates whether audio channel at spatial location 0 is currently in use in audio network
	[15+28*i*][6:4]	ACH_FORMAT[0][2:0]	Audio type used by audio channel at spatial location 0: 0: unused audio channel 1: PCM16 2: PCME24 4: SLAC 5: PCMLF
	[15+28*i*][3]	ACH_ACTIVE[1]	Flag that indicates whether audio channel at spatial location 1 is currently in use in audio network
	[15+28*i*][2:0]	ACH_FORMAT[1][2:0]	Audio type used by audio channel at spatial location 1
	…		
	[22+28*i*][3]	ACH_ACTIVE[15]	Flag that indicates whether audio channel at spatial location 15 is currently in use in audio network
	[22+28*i*][2:0]	ACH_FORMAT[15][2:0]	Audio type used by audio channel at spatial location 15
	[23+28*i*]	RSSI[7:0]	Averaged RSSI of packets received from master during scan operation (signed 2's complement, 0 ➔ ~0 dBm)
	[24+28*i*:25+28*i*][15:12]	–	Reserved, ignore value
	[24+28*i*:25+28*i*][11:0]	SAMPLE_RATE[11:0]	Sample rate (in increments of 25 Hz) as reported by master
	[26+28*i*:27+28*i*]	–	Reserved, ignore value

Command	NWM_DO_SCAN		
	[15:12] [26+28*i*:27+28*i*] [11:0]	LATENCY[11:0]	Audio latency in audio network (in samples).
Related events			
Usage limitations	Only available in host-controlled protocol slaves		
Notes	Depending on the number of networks found, the number of returned bytes will be N * 28, where N is between 0 and SCAN_MAX. The parameters to this command were changed in FW 1.3.0 (two bytes were added).		

Command	NWM_DO_JOIN		
Description	Used by protocol slaves to join a specific PurePath Wireless network or the first found that matches the specified criteria. The command is also used to leave a network.		
SPI Operations	CMD_REQ(0x09, 18, pars, sw)		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0:1][15]	–	Reserved, write 0
	[0:1][14:0]	JOIN_TO[11:0]	Timeout in increments of 10 ms for joining network.
	[2:5]	DEVICE_ID[31:0]	Network ID of network to join (device ID of master) A value of 0x00000000 means leave network if currently connected A value of 0xFFFFFFFF means join any network where master is currently attempting to pair with a new slave
	[6:9]	MANF_ID[31:0]	Manufacturer ID filtering criteria. A value of 0 means ignore manufacturer ID of master, any other value requires that the protocol master has a matching manufacturer ID.
	[10:13] [14:17]	PROD_ID_MASK[31:0] PROD_ID_REF[31:0]	Product/family ID mask and reference value to be used as filtering criteria for whether to join network. Only networks where PROD_ID&PROD_ID_MASK == PROD_ID_REF&PROD_ID_MASK for masters with PROD_ID will be joined. Thus a PROD_ID_MASK value of 0 will disable filtering based on product ID.
Related events	EVT_NWK_CHG – will be signaled when a network connection is lost – or if this DO_JOIN command fails		
Usage limitations	Only available on host-controlled protocol slaves		
Notes	The host controller should use NWM_GET_STATUS to determine the outcome of the command and to store the network ID for future pairing operations if required. The command will use ~500 ms to scan the entire RF channel space once. Using a JOIN_TO value below 50 (= 500 ms) is not recommended.		

Command	NWM_GET_STATUS		
Description	Returns status about current audio network status and other nodes in network.		
SPI Operations	CMD_REQ(0x0A, 0, sw) On protocol slave: READBC(sw, nBytes, data)		
(data)	**[Byte][bit] index**	**Field**	**Description**
			nBytes will always be 30 when connected and 0 otherwise.
	[0:3]	DEVICE_ID[31:0]	Network ID of current network (device ID of master)
	[4:7]	MANF_ID[31:0]	Manufacturer ID of protocol master
	[8:11]	PROD_ID [31:0]	Product ID of protocol master
	[12][7:5]	WPM_PM_STATE	Protocol master's assumed power state (possible values): 5: ACTIVE (1+ logical channel subscriptions active) 4: LOW_POWER (0 logical channel subscriptions active) 2: NWK_STANDBY (network standby signaling active)
	[12][4]	WPM_DSC_EN	Indicates that master supports data side channel communication
	[12][3]	WPM_MFCT_FILT	Indicates that master only accepts protocol slaves with matching manufacturer ID into network
	[12][2]	WPM_PAIR_SIGNAL	Indicates that master is signaling that it is attempting to pair with a new slave
	[12][1]	WPM_ALLOWS_JOIN	Indicates that master can accept more protocol slaves into

Command	NWM_GET_STATUS		
			network
	[12][0]	–	Reserved, ignore value
	[13:14]	ACH_SUPPORT[15:0]	Bitmask indicating which audio channels the protocol master supports. The bit indexes correspond to the logical audio channel indexes defined for NWM_ACH_SET_USAGE.
	[15][7]	ACH_ACTIVE[0]	Flag that indicates whether audio channel at spatial location 0 is currently in use in audio network
	[15][6:4]	ACH_FORMAT[0][2:0]	Audio type used by audio channel at spatial location 0 0: unused audio channel 1: PCM16 2: PCME24 4: SLAC 5: PCMLF
	[15][3]	ACH_ACTIVE[1]	Flag that indicates whether audio channel at spatial location 1 is currently in use in audio network
	[15][2:0]	ACH_FORMAT[1][2:0]	Audio type used by audio channel at spatial location 1
			…
	[22][3]	ACH_ACTIVE[15]	Flag that indicates whether audio channel at spatial location 15 is currently in use in audio network
	[22][2:0]	ACH_FORMAT[15][2:0]	Audio type used by audio channel at spatial location 15
	[23]	RSSI[7:0]	Averaged RSSI of recent packets received from master (signed 2s complement, 0 → ~0 dBm)
	[24:25][15:12]	–	Reserved, ignore value
	[24:25][11:0]	SAMPLE_RATE[11:0]	Sample rate (in increments of 25 Hz) as reported by master
	[26:27][15:12]	NWK_STATUS	Current network status of protocol slave (this node): 0: No network connection 8/13: Connected to network other: Searching/tracking/joining network
	[26:27][11:0]	LATENCY[11:0]	Audio latency in audio network (in samples).
	[28:29]	ACH_USED[15:0]	Bitmask providing information about which audio channels are currently used by slave. The bit indexes correspond to the logical audio channel indexes defined for NWM_ACH_SET_USAGE.
	On protocol master: `READBC(sw, nBytes, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	[0][7:4]	–	Reserved, ignore value
	[0][3:0]	NWK_STATUS	Current network status of protocol master: 0: No network maintained other: Maintaining network
	[1:2][15:12]	–	Reserved, ignore value
	[1:2][11:0]	SAMPLE_RATE[11:0]	Current sample rate (in increments of 25 Hz)
	[3:4]	ACH_USED[15:0]	Bitmask indicating which audio channels is consumed/produced by any protocol slaves.
	For each connected slave, *i* =[0, (nBytes-5)/16-1]. nBytes=5 means no protocol slaves connected.		
	[5+16*i*:8+16*i*]	DEVICE_ID[31:0]	Device ID of slave
	[9+16*i*:12+16*i*]	MANF_ID[31:0]	Manufacturer ID reported by protocol slave
	[13+16*i*:16+16*i*]	PROD_ID[31:0]	Product/family ID reported by protocol slave
	[17+16*i*:18+16*i*]	ACH_USED[15:0]	Bitmask indicating which audio channels the protocol slave is consuming/producing. The bit indexes correspond to the logical audio channel indexes defined for NWM_ACH_SET_USAGE.
	[19+16*i*]	–	Reserved, ignore value
	[20+16*i*][7:4]	–	Reserved, ignore value
	[20+16i][3:1]	SLAVE_SHORT_ID	Short ID for referencing a protocol slave (1-7)
	[20+16*i*][0]	WPS_DSC_EN	If set, indicates that protocol slave supports a data side-channel.
Related events	–		
Usage limitations	Available on both host-controlled protocol masters and slaves but returned data must be interpreted differently		
Notes			

Command	NWM_ACH_SET_USAGE
Description	Used on protocol slaves to define how logical audio channels supported in audio network should be mapped to local audio interface channels as defined by the PurePath Wireless Configurator. This command is used to force the protocol slave to start/stop producing/consuming certain audio channels.
SPI Operations	`CMD_REQ(0x0B, 16, pars, sw)`

(pars)	**[Byte][bit] index** / **Logical audio channel** / **Description**
	`[0]` `FRONT_PRIMARY_LEFT` — 0 – MAX: Configures the local audio interface channel for the logical audio channel. MAX is the maximum number of local audio interface channels given by the setup in the PurePath Wireless Configurator.
	`[1]` `FRONT_PRIMARY_RIGHT`
	`[2]` `REAR_SECONDARY_LEFT`
	`[3]` `REAR_SECONDARY_RIGHT`
	`[4]` `FRONT_CENTER`
	`[5]` `SUBWOOFER` — 0xFF: Indicates the channel is not in use.
	`[6]` `SIDE_LEFT` — Others: Not allowed
	`[7]` `SIDE_RIGHT`
	`[8]` `USER_DEFINED_0`
	`[9]` `USER_DEFINED_1`
	`[10]` `INPUT_LEFT`
	`[11]` `INPUT_RIGHT`
	`[12]` `MICROPHONE_0`
	`[13]` `MICROPHONE_1`
	`[14]` `MICROPHONE_2`
	`[15]` `MICROPHONE_3`

Related events	-
Usage limitations	Only available on host-controlled protocol slaves Command only has effect if the protocol slave is currently connected to a network
Notes	Upon joining a network no audio channels are used and this command must be used to start audio streaming. The host controller can use the NWM_GET_STATUS command to determine which audio channels are available in audio network and which the protocol slave is currently using. This command can be used to dynamically change which channels a protocol slave is consuming/producing during a network connection. The audio channels available in a network are statically configured in PurePath Wireless Configurator for a protocol master. Mono slave example: If the master produces and distributes the subwoofer channel, the protocol slave can use NWM_ACH_SET_USAGE with `SUBWOOFER`=0x00 and Others=0xFF to start receiving the channel and output it on the left channel of the audio interface. Stereo slave example: If the master produces and distributes the front left and front right channel, the slave can use NWM_ACH_SET_USAGE with `FRONT_PRIMARY_LEFT` = 0x00, `FRONT_PRIMARY_RIGHT`=0x01 and Others=0xFF to start receiving these channels and output them on the left and right channels of the audio interface.

Command	NWM_CONTROL_ENABLE
Description	Used on protocol masters to enable/disable formation/maintenance of the audio network. Advertisement for pairing will be reset when disabling the network.
SPI Operations	`CMD_REQ(0x0C, 2, pars, sw)`
(pars)	**[Byte][bit] index** / **Field** / **Description** `[0]` – Reserved, write 0 `[1][7:1]` – Reserved, write 0 `[1][0]` `WM_ENABLE` Enable (1) or disable (0) formation/maintenance of network.
Related events	-
Usage limitations	Only available on host-controlled protocol masters
Notes	

Command	NWM_CONTROL_SIGNAL		
Description	Used on protocol masters to enable/disable pairing of new protocol slaves to the network. The advertisement for pairing is reset when a protocol slave joins or a network is disabled (i.e. with the NWM_CONTROL_ENABLE command).		
SPI Operations	`CMD_REQ(0x0D, 2, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0]`	–	Reserved, write 0
	`[1][7:1]`	–	Reserved, write 0
	`[1][0]`	`WM_PAIR_SIGNAL`	When set the protocol master indicates in every packet that it is trying to pair with a slave.
Related events	-		
Usage limitations	Only available on host-controlled protocol masters		
Notes	The `WM_PAIR_SIGNAL` field reflects what is reported by `NWM_DO_SCAN` commands performed on a protocol slave. If the `NWM_DO_JOIN` command is run on a protocol slave with parameter `DEVICE_ID`=0xFFFFFFFF the slave will only join this master if `WM_PAIR_SIGNAL` is set. The `WM_PAIR_SIGNAL` is automatically cleared when a protocol slave joins the network.		

Command	NWM_SET_RF_CH_MASK		
Description	Used on protocol masters to set the currently used or to be used RF channel mask or to enable/disable the radio. A valid RF channel mask contains 6 to 18 RF channels. Setting an RF channel mask containing 0 to 5 RF channels suspends network maintenance. Switching back to a valid RF channel mask re-enables network maintenance. RF channels masked out at configuration time cannot be enabled using the NWM_SET_RC_CH_MASK command.		
SPI Operations	`CMD_REQ(0x0E, 4, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0:3][31:19]`	–	Reserved, write 0
	`[0:3][18:1]`	`RF_CH_MASK[18:1]`	Enable (1) or disable (0) RF channels 1 through 18
	`[0:3][0]`	–	Reserved, write 0
Related events	-		
Usage limitations	Only available on protocol masters		
Notes	When changing the RF channel mask, the transition to the new channel set happens gradually to avoid audio drop-outs and loss of network connection. The time required to make the switch is non-deterministic, but the transition can be monitored using the PS_RF_STATS command.		

B.3.4 Remote Control Commands

Command	RC_SET_DATA
Description	Used on protocol slaves to communicate either pre-defined or custom-defined remote control information to the protocol master. Protocol master and slaves must use the same scheme (pre- or custom-defined); otherwise the remote control input will be ignored. Note that keyboard/button input may be lost if held active by RC_SET_DATA for a too short period of time. Note also that unwanted "double-clicks" may occur if keyboard/button input is held active for more than 90 ms. **For pre-defined input:** Three types of information can be transmitted: - Remote control command codes (play/pause, skip etc.), as specified in Table 16. - Keyboard key codes, as specified in the Keyboard/Keypad page of USB HID Usage Tables 1.11. - Mouse buttons and position (two-axis). To send remote control and keyboard codes, add the active codes to `CMD_STATE` and `KEYB_STATE` and set `CMD_COUNT` or `KEYB_COUNT` to match the number of valid/active codes. To send mouse information, `EXT_SEL` must be set to 1. To send a mouse movement, add or subtract the new movement from the last communicated `MOUSE_POSITION_X` and `MOUSE_POSITION_Y`. **For custom-defined input:** The information provided, up to 12 bytes, is user-specified, and must correspond with the protocol master's interpretation of the data. For custom-defined USB HID reports, the format is defined by the user-written HID report definition file.
SPI Operations	`CMD_REQ(0x2D, 13, pars, sw)` For pre-defined input:

(pars)

[Byte][bit] index	Field	Description
[0][7:6]	EXT_SEL	Specifies contents of bytes [8:12]: 0 = None 1 = Mouse 2/3 = Reserved, do not use
[0][5:3]	KC (KEYB_COUNT)	Number of active bytes in KEYB_STATE
[0][2:0]	CC (CMD_COUNT)	Number of active bytes in CMD_STATE
[1:KC]	KEYB_STATE[0:KC-1]	KEYB_STATE[0] is modifier bits (ctrl, shift etc.) KEYB_STATE[1..KC-1] are US keyboard key code(s) Field is not present when KEYB_COUNT=0
[1+KC:KC+CC]	CMD_STATE[0:CC-1]	Active remote control command ID(s) Field is not present when CMD_COUNT=0
[1+KC+CC:7]	FILL[]	Filler bytes, write 0x00
[8]	MOUSE_BUTTONS	EXT_SEL = 1: Bit vector specifying pressed mouse buttons (left, middle, right etc.)
[9:10]	MOUSE_POSITION_X	EXT_SEL = 1: Mouse horizontal (X) position
[11:12]	MOUSE_POSITION_Y	EXT_SEL = 1: Mouse vertical (Y) position

For custom-defined input:

(pars)

[Byte][bit] index	Field	Description
[0][7:6]	DATA_SEL	Specifies contents of bytes [1:12]: 0/1/3 = Reserved, do not use 2 = HID report raw data or proprietary data
[0][5:0]	ZERO	Reserved, write 0
[1:12]	DATA[0:11]	DATA_SEL = 2: HID report raw data or other proprietary remote control information

Usage limitations	Only available in host-controlled operation on protocol slave. `KEYB_COUNT` + `CMD_COUNT` must be 7 or less.
Notes	It is not necessary to release/reset the keyboard/command states before leaving a network. The protocol master's timeout will ensure that keys do not become sticky, and for custom-defined input all data will return to zero. Note that for keyboard codes, modifier bits are always included first. So, for example, in the case where only one letter is active, `KEYB_COUNT` should be 2 (the modifier byte and the byte for the letter). `KEYB_COUNT=1` means that only the modifier bits will be included (e.g. only shift key is pressed). `EXT_SEL`, `MOUSE_BUTTONS`, `MOUSE_POSITION_X` and `MOUSE_POSITION_Y` were added in FW 1.4.0. Option for custom-defined remote control information was added in FW 1.4.1.

<table>
<tr><th>Command</th><th>RC_GET_DATA</th></tr>
<tr><td>Description</td><td>Used on protocol master to retrieve the remote control button/keyboard states/mouse button and position for the requested protocol slave. For details, see the RC_SET_DATA command.

The remote control information is not interpreted by the CC85xx. The host processor is responsible for combining information from multiple protocol slaves in sensible way.</td></tr>
<tr><td>SPI Operations</td><td>CMD_REQ(0x2E, 1, pars, sw)</td></tr>
<tr><td>(pars)</td><td>
<table>
<tr><th>[Byte][bit] index</th><th>Field</th><th>Description</th></tr>
<tr><td>[0]</td><td>SLAVE_SHORT_ID</td><td>Index of the protocol slave slot used (1-7), retrieved in the NWM_GET_STATUS command</td></tr>
</table>
</td></tr>
<tr><td></td><td>READ(13, sw, data)</td></tr>
<tr><td>(data)</td><td>For pre-defined input:
<table>
<tr><th>[Byte][bit] index</th><th>Field</th><th>Description</th></tr>
<tr><td>[0][7:6]</td><td>EXT_SEL</td><td>Specifies contents of bytes [8:12]:
0 = None
1 = Mouse
2/3 = Reserved, ignore all other fields if used</td></tr>
<tr><td>[0][5:3]</td><td>KC (KEYB_COUNT)</td><td>Number of active bytes in KEYB_STATE</td></tr>
<tr><td>[0][2:0]</td><td>CC (CMD_COUNT)</td><td>Number of active bytes in CMD_STATE</td></tr>
<tr><td>[1:KC]</td><td>KEYB_STATE[0:KC-1]</td><td>KEYB_STATE[0] is modifier bits (ctrl, shift etc.)
KEYB_STATE[1..KC-1] are US keyboard key code(s)
Field is not present when KEYB_COUNT=0</td></tr>
<tr><td>[1+KC:KC+CC]</td><td>CMD_STATE[0:CC-1]</td><td>Active remote control command ID(s)
Field is not present when CMD_COUNT=0</td></tr>
<tr><td>[1+KC+CC:7]</td><td>FILL[]</td><td>Filler bytes, ignore values</td></tr>
<tr><td>[8]</td><td>MOUSE_BUTTONS</td><td>EXT_SEL = 1: Bit vector specifying pressed mouse buttons (left, middle, right etc.)</td></tr>
<tr><td>[9:10]</td><td>MOUSE_POSITION_X</td><td>EXT_SEL = 1: Mouse horizontal (X) position</td></tr>
<tr><td>[11:12]</td><td>MOUSE_POSITION_Y</td><td>EXT_SEL = 1: Mouse vertical (Y) position</td></tr>
</table>
</td></tr>
<tr><td>(data)</td><td>For custom-defined input:
<table>
<tr><th>[Byte][bit] index</th><th>Field</th><th>Description</th></tr>
<tr><td>[0][7:6]</td><td>DATA_SEL</td><td>Specifies contents of bytes [1:12]:
0/1/3 = Reserved, ignore DATA if used
2 = HID report raw data or other proprietary remote control information</td></tr>
<tr><td>[0][5:0]</td><td>ZERO</td><td>Reserved, ignore DATA if non-zero</td></tr>
<tr><td>[1:12]</td><td>DATA[0:11]</td><td>DATA_SEL = 2: HID report raw data or other proprietary remote control information</td></tr>
</table>
</td></tr>
<tr><td>Usage limitations</td><td>Only available in host-controlled operation on protocol master</td></tr>
<tr><td>Notes</td><td>If KEYB_COUNT is 0, no KEYB_STATE data will be returned. This signals that no keyboard keys are active.
If CMD_COUNT is 0, no CMD_STATE data will be returned. This signals that no commands are active.

The protocol master will set all remote states to inactive if remote control state information is not received from the protocol slave for more than 100ms. For the custom-defined scheme, DATA_SEL and DATA are reset to all zero.

EXT_SEL, MOUSE_BUTTONS, MOUSE_POSITION_X and MOUSE_POSITION_Y were added in FW1.4.0.
Option for custom-defined remote control information was added in FW 1.4.1.</td></tr>
</table>

B.3.5 Data Side-Channel Commands

Command	DSC_TX_DATAGRAM		
Description	Queues data side-channel datagram for transmission if space is available		
SPI Operations	`CMD_REQ(0x04, 5, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0][7:1]`	–	Reserved, write 0.
	`[0][0]`	`CONN_RESET`	Resets/opens a data side-channel connection. Set this flag in the first datagram after a network connection is established.
	`[1:4]`	`ADDR`	32b device ID of datagram destination.
	`WRITE(nBytes, sw, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0:nBytes-1]`	`DATA[0:nBytes-1]`	User-specified datagram payload (1-32 bytes)
Related events	`EVT_DSC_TX_AVAIL` `EVT_DSC_RESET`		
Usage limitations	Only available in host-controlled operation.		
Notes	Use the output from NWM_GET_STATUS to find potential data side-channel peers. When a network connection is established or re-established, the data side-channel will be closed. To open a data side-channel connection, a datagram with `CONN_RESET`=1 must be submitted to the TX queue. This must be done for each master-slave network connection, and for each data side-channel data direction. While the data side-channel is closed, datagrams submitted with `CONN_RESET`=0 will be dropped silently. This command shall only be executed when `EVT_DSC_TX_AVAIL` is high. Datagrams submitted while `EVT_DSC_TX_AVAIL` is low may be dropped, and if this happens, `EVT_SPI_ERR` will be set to indicate error. Datagrams with invalid destination address (i.e. the destination is not currently part of the network, or does not support the data side-channel) will be dropped silently. Note that datagrams will not be removed from the TX queue while network maintenance is disabled, and may (if the `CONN_RESET` flag is set) be transmitted when network maintenance is resumed. Note also that datagrams are not removed from the TX queue until it has been successfully transferred. This means that a receiver who does not confirm reception will effectively block the data side-channel for other destinations. Once a receiver is ejected from the network, its datagrams are silently discarded, and the data side-channel will proceed processing other destinations from the TX queue. Queuing a datagram with a payload length of 0 bytes, will give an EVT_SPI_ERROR		

Command	DSC_RX_DATAGRAM		
Description	Gets side-channel datagram from the receive queue, if available.		

SPI Operations	`CMD_REQ(0x05, 0, sw)`
	
	`READBC(sw, nBytes, data)`		

(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0][7:1]`	–	Reserved, ignore value.
	`[0][0]`	`CONN_RESET`	Indicates the first datagram in a data side-channel connection/network connection.
	`[1:4]`	`ADDR`	Protocol master: 32b device ID of datagram source Protocol slave: Own 32b device ID (Value 0 means that reception queue is empty)
	`[5:nBytes-1]`	`DATA[0:nBytes-5]`	User-specified datagram payload (1-32 bytes) (Length 0 means that reception queue is empty)
Related events	`EVT_DSC_RX_AVAIL`		
Usage limitations	Only available in host-controlled operation		
Notes	Datagrams must be read from the RX queue as quickly as possible to avoid blocking the transmitter's TX queue (if more datagrams to the same destination are pending). If the RX queue is empty, a dummy datagram without payload will be returned, i.e. `nBytes=5`. Note that if multiple DSC_RX_DATAGRAM commands are executed back-to-back, the `EVT_DSC_RX_AVAIL` flag can temporarily be high, even if the RX queue actually is empty.		

B.3.6 Power Management Commands

Command	PM_SET_STATE		
Description	Sets the device power state, for power reduction control and power-down.		
SPI Operations	`CMD_REQ(0x1C, 1, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0]	PM_STATE	Set the desired power state of the device: 6,7 - Reserved for future extensions, do not use 5: ACTIVE 4: LOW-POWER - force audio device into low-power state 3: LOCAL STANDBY - force audio device into inactive state 2: NETWORK STANDBY - force audio device into configurable state, activate network standby signaling (protocol master only) 1 - Reserved for future extensions, do not use 0: OFF - power down the device
Related events	EVT_PS_CHG		
Usage limitations	Only available in host-controlled operation		
Notes	The effective power state may be lower than the one specified here, e.g. protocol slaves may be limited to the NETWORK STANDBY state due to network standby signaling from the protocol master. The device will not go into the OFF power state until the CSN pin is deasserted. The device will be reset and enter the ACTIVE state by pulling CSN low after turning the device OFF with this command. The device should not be held in reset after entering the OFF power state, as this will re-enable the digital regulator and the crystal oscillator.		

Command	PM_GET_DATA		
Description	Returns power management related information. Using this information, the host processor can control the power state (using the PM_SET_STATE command) to perform automatic power-down and/or automatic power reduction.		

SPI Operations	`CMD_REQ(0x1D, 0, sw)`
	
	`READBC(sw, nBytes, data)`		

(data)	**[Byte][bit] index**	**Field**	**Description**
	[0:3]	IN_SILENCE_TIME	Period of time with analog or digital silence on all locally input audio channels, in increments of 10 ms.
	[4:7]	OUT_SILENCE_TIME	Period of time with digital silence (zero amplitude) on all locally output audio channels, in increments of 10 ms.
	[8:11]	NWK_INACTIVITY_TIME	Period of time without any network connection(s), in increments of 10 ms.
	[12:13]	VBAT_VOLTAGE	Last measured battery voltage in millivolts, provided that VBAT monitoring is enabled at configuration time. The value is updated 4 times per second.
Related events			
Usage limitations	Only available in host-controlled operation		
Notes	This command was changed to reporting audio input and output silence independently in FW 1.4.0.		

B.3.7 Volume Control Commands

Command	VC_SET_VOLUME		
Description	Configures slave global/remote (protocol master only) or local input or output volume		
SPI Operations	`CMD_REQ(0x17, 4, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0][7:2]	-	Reserved, write 0
	[0][1]	IS_IN_VOL	0: Output volume, 1:Input volume
	[0][0]	IS_LOCAL	0: Remote, 1:Local
	[1][7:6]	MUTE_OP	0: None 1: Mute 2: Unmute 3: Toggle
	[1][5:4]	SET_OP	0: None 1: Absolute 2: Relative 3: Logical channel offset
	[1][3:0]	LOG_CHANNEL	Logical channel (only relevant when SET_OP=3)
	[2:3]	VOLUME	Volume setting in units of 0.125 dB (signed 2's complement)
Related events	EVT_VOL_CHG		
Usage limitations	Only available in host-controlled operation		
Notes	All devices are muted upon power-up and must be unmuted in order for volume settings to be effective.		

Command	VC_GET_VOLUME		
Description	On protocol master: Returns slave global/remote or local input or output volume. On protocol slave: Returns local input, output volume, mono channel offset (for mono-channel configurations) or logical channel offset (for multi-channel configurations). For multi-channel protocol slaves, the device configuration determines whether the logical channel offsets communicated by the protocol master are used or ignored (default). The host controller is responsible for saturating the sum of volume setting and channel offset. Muting shall occur if the volume setting corresponds to -128.0 dB, regardless of the channel offset value.		
SPI Operations	`CMD_REQ(0x16, 1, pars, sw)`		
	On protocol master:		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0][7:2]	-	Reserved, write 0
	[0][1]	IS_IN_VOL	0: Output volume, 1: Input volume
	[0][0]	IS_LOCAL	0: Remote, 1:Local
	On protocol slave:		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0][7:6]	-	Reserved, write 0
	[0][5:2]	LOG_CHANNEL	Logical channel (only relevant when streaming more than one logical channel and configured not to ignore logical channel offsets). Note that up-to-date channel offsets will only be returned for currently streamed logical channels.
	[0][1]	IS_IN_VOL	0: Output volume, 1: Input volume
	[0][0]	IS_CHANNEL_OFFSET	0: Volume setting, 1: Mono/logical channel offset
	`READ(2, sw, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	[0:1]	VOLUME	Volume/channel offset in units of 0.125 dB (signed 2's complement)
Related events	EVT_VOL_CHG		
Usage limitations	Only available in host-controlled operation		
Notes			

B.3.8 RF and Audio Statistics Commands

Command	PS_AUDIO_STATS		
Description	Requests and returns audio statistics gathered since the last PS_AUDIO_STATS command or chip reset. The counter values for processed/concealed/muted samples and mute events are only valid for locally consumed audio channels. The counters are reset to zero after running the command. If not polled regularly, the counters will saturate (at 0xFFFFFFFF for 32-bit values and 0xFFFF for 16-bit values). The peak and mean values are valid for both produced and consumed audio channels.		

SPI Operations	`CMD_REQ(0x11, 0, sw)`
	
	`READBC(sw, nBytes, data)`		

(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0:3]`	`SMPL_PROCESS_COUNT16`	Number of processed samples/16 contributing to the counter values below.
	`[4:7]`	`SMPL_CONCEAL_COUNT16`	Number of samples/16 where error concealment was active (due to missing odd audio slices for PCM16/PCME24).
	`[8:11]`	`SMPL_MUTED_COUNT16`	Number of muted samples/16 where muting was active (due to missing critical audio slices)
	`[12:13]`	`MUTE_EVENT_COUNT`	Number of mute events, i.e. number of transitions from unmuted to muted audio outputs.
	`[14]`	`CH_COUNT`	Number of channels in the peak and mean value statistics output
	`[15]`	–	Reserved, ignore value
		For each channel, *i* =[0, ((nBytes-16)/4)-1].	
	`[16+4i:17+4i]`	`CH_PEAK_VALUE`	Peak absolute value
	`[18+4i:19+4i]`	`CH_MEAN_VALUE`	Mean absolute value (filtered)
Related events	-		
Usage limitations	Available in host-controlled and autonomous operation		
Notes	For the period between chip reset and initial audio streaming, there are muted samples, but no mute event. For the MICROPHONE_0 to MICROPHONE_3 logical audio channels, mute events are counted separately for each.		

Command	PS_RF_STATS		
Description	Requests and returns RF statistics gathered since the last PS_RF_STATS command or chip reset. All counters are reset to zero after the command.		

SPI Operations	`CMD_REQ(0x10, 0, sw)`
	
	`READBC(sw, nBytes, data)`		

(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0:3]`	`TIMESLOT_COUNT`	Number of timeslots contributing to the statistics output
	`[4:7]`	`RX_PKT_COUNT`	Number of packet receptions attempted
	`[8:11]`	`RX_PKT_FAIL_COUNT`	Number of packet receptions that failed
	`[12:15]`	`RX_SLICE_COUNT`	Slave: Number of slices received Master: Number of slices transmitted
	`[16:19]`	`RX_SLICE_ERR_COUNT`	Slave: Number of slices received with errors Master: Reserved, ignore value
	`[20]`	`NWK_JOIN_COUNT`	Slave: Number of successful network join Master: Number of successful slave join
	`[21]`	`NWK_DROP_COUNT`	Slave: Number of network drops Master: Number of slave drops
	`[22:23]`	`AFH_SWAP_COUNT`	Slave: Reserved, ignore value Master: Number of times the adaptive frequency hopping algorithm swapped out an active RF channel.
		For each RF channel *i* = [0,19]	
	`[24+2i:25+2i]`	`AFH_CH_USAGE_COUNT`	Number of times RF channel *i* is used
Related events	-		
Usage limitations	Available on host-controlled and autonomous devices		
Notes			

B.3.9 Calibration Commands

Command	CAL_SET_DATA		
Description	One-shot command that modifies the configured target TX output power for audio network operation. The modification is stored in on-chip flash memory. This command can be used in the production test of the final application to attain more accurate output power after stepping through various `TX_POW` settings with the RFT_TXTST_PN command and measuring the actual output power. The actual output power may differ slightly from the target output power and will also vary with temperature and frequency. Output power is measured in chip production at 2442 MHz and for nominal operating conditions. The CAL_SET_DATA command must be used after the flash programming procedure for the application firmware image. The command alters the programmed image, and causes future BL_FLASH_VERIFY commands to indicate error. Hence the CAL_SET_DATA command should be verified using the CAL_GET_DATA command.		
SPI Operations	`CMD_REQ(0x28, 5, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0]`	–	Write key 0x5F, if any other value command fails
	`[1:3]`	–	Reserved, write 0
	`[4]`	`TX_POW`	Target TX power in dBm (2's complement)
Related events	-		
Usage limitations	Available in host-controlled and autonomous operation. This command can only be used once after (re)programming the device		
Notes	Additional fields may be added in future FW releases, but existing fields will retain their index.		

Command	CAL_GET_DATA		
Description	Reads calibration data		

SPI Operations	`CMD_REQ(0x29, 0, sw)`
	
	`READ(4, sw, data`		

(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0][7]`	`NOT_VALID`	1 if never written, 0 if CAL_SET_DATA has been used
	`[0][6:0]`	–	Reserved, ignore value
	`[1:2]`	–	Reserved, ignore value
	`[3]`	`TX_POW`	Target TX power in dBm (2's complement)
Related events	-		
Usage limitations	Available in host-controlled and autonomous operation.		
Notes	Additional fields may be added in future FW releases, but existing fields will retain their index.		

B.3.10 Utility Commands

Command	IO_GET_PIN_VAL		
Description	Polls the current value of I/O pins GIO1 through GIO15. This can provide small microcontrollers with extra input pins.		

SPI Operations	`CMD_REQ(0x2A, 0, sw)`
	
	`READ(4, sw, data)`		

(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0]`	–	Reserved, ignore value
	`[1]`	–	Reserved, ignore value
	`[2][7]`	–	Reserved, ignore value
	`[2][6]`	`GIO15_VAL`	Value of GIO15
	`[2][5]`	`GIO14_VAL`	Value of GIO14
			…
	`[2][0]`	`GIO9_VAL`	Value of GIO9
	`[3][7]`	`GIO8_VAL`	Value of GIO8
			…
	`[3][0]`	`GIO1_VAL`	Value of GIO1
Related events	-		
Usage limitations	Only available in host-controlled operation.		
Notes	All 15 I/O pins are sampled concurrently; however the exact sampling time is non-deterministic and may occur at any time between the CMD_REQ operation and the CMDREQ_RDY signal going high.		

Command	NVS_GET_DATA
Description	Reads data from non-volatile storage implemented in the CC85xx internal flash memory. This command can be used • In host-controlled operation: To read back data previously written to non-volatile storage by NVS_SET_DATA. • In autonomous operation (protocol slave only): To verify that pre-pairing using NVS_SET_DATA has succeeded.
SPI Operations	`CMD_REQ(0x2B, 1, pars, sw)`
(pars)	**[Byte][bit] index** / **Field** / **Description** `[0][7:2]` / – / Reserved, write 0 `[0][1:0]` / `SLOT_INDEX` / Index of 32-bit data slot to read. • Autonomous protocol slave: 0 = network ID • Host-controlled protocol master/slave: 0/1/2/3 = custom data
	`READ(4, sw, data)`
(data)	**[Byte][bit] index** / **Field** / **Description** `[0:3]` / `DATA` / Read data (The default value is 0x00000000 for host-controlled operation, and configured network ID for autonomous operation)
Related events	-
Usage limitations	Available in host-controlled operation (protocol master and slave) and autonomous operation (protocol slave only), with different interpretation of the `SLOT_INDEX` field.
Notes	

Command	NVS_SET_DATA
Description	Writes data to non-volatile storage implemented in the CC85xx internal flash memory. This command can be used: • In host-controlled operation: To emulate an external EEPROM able to store four 32-bit words (network ID, volume settings etc.). • In autonomous operation (protocol slave only): For pre-pairing, i.e. to pre-program the network ID during production. The protocol master's device ID (which becomes the network ID) can be read out during production using DI_GET_DEVICE_INFO.
SPI Operations	`CMD_REQ(0x2C, 5, pars, sw)`
(pars)	**[Byte][bit] index** / **Field** / **Description** `[0][7:2]` / – / Reserved, write 0 `[0][1:0]` / `SLOT_INDEX` / Index of 32-bit data slot to write - Autonomous protocol slave: 0 = network ID - Host-controlled protocol master/slave: 0/1/2/3 = custom data `[1:4]` / `DATA` / Data to be written NOTE: Writing 0xFFFFFFFF to slot 0 is not supported.
Related events	-
Usage limitations	Available in host-controlled operation (protocol master and slave) and autonomous operation (protocol slave only), with different interpretation of the `SLOT_INDEX` field.
Notes	This command must not be executed while maintaining a network or while the power supply is unstable. Every 128th NVS_SET_DATA command will take an additional ~25 ms to complete.

B.3.11 RF Test Commands

Command	RFT_TXTST_CW		
Description	Outputs a continuous wave RF signal at a specific frequency.		
SPI Operations	`CMD_REQ(0x15, 3, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0][7:1]`	–	Reserved, write 0
	`[0][0]`	`ENABLE`	Enable/disable CW transmission
	`[1]`	`FREQ_OFFSET`	RF frequency for CW output (2318 + `FREQ_OFFSET`) MHz.
	`[2]`	`TX_POW`	Target TX power in dBm (2's complement)
Related events	-		
Usage limitations	Only available in production test operation		
Notes	A device reset should be performed before and after using this command.		

Command	RFT_TXTST_PN		
Description	Outputs a pseudo-random modulated RF signal at a specific frequency		
SPI Operations	`CMD_REQ(0x14, 3, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0][7:1]`	–	Reserved, write 0
	`[0][0]`	`ENABLE`	Enable/disable PN transmission
	`[1]`	`FREQ_OFFSET`	RF frequency (2318 + `FREQ_OFFSET`) MHz.
	`[2]`	`TX_POW`	Target TX power in dBm (2's complement)
Related events	-		
Usage limitations	Only available in production test operation		
Notes	A device reset should be performed before and after using this command.		

Command	RFT_RXTST_CONT		
Description	Enables continuous reception at a specific frequency.		
SPI Operations	`CMD_REQ(0x25, 2, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0][7:1]`	–	Reserved, write 0
	`[0][0]`	`ENABLE`	Enable/disable receiver
	`[1]`	`FREQ_OFFSET`	RF frequency (2318 + `FREQ_OFFSET`) MHz
Related events	-		
Usage limitations	Only available in production test operation		
Notes	A device reset should be performed before and after using this command.		

Command	RFT_RXTST_RSSI		
Description	Measures the RSSI at a specific frequency		
SPI Operations	`CMD_REQ(0x26, 1, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0]`	`FREQ_OFFSET`	RF frequency (2318 + `FREQ_OFFSET`) MHz
	`READ(1, sw, data)`		
(data)	**[Byte][bit] index**	**Field**	**Description**
	`[0]`	`RSSI`	Measured RSSI value in dBm (signed 2's complement)
Related events	-		
Usage limitations	Only available in production test operation		
Notes	A device reset should be performed before and after using this command. When used in conjunction with RFT_TXTST_CW, the `FREQ_OFFSET` should be offset with +/- 1 MHz vs. the one used in RFT_TXTST_CW.		

Command	RFT_TXPER		
Description	Runs the transmitter side of the packet error rate test. The test uses a packet format similar to the CC85xx radio protocol, with pseudo-random payload data. Each packet contains a header of 16 bytes (incl. individual checksum) and 16 data blocks (corresponding to PCM16 audio slices) of 36 bytes each (incl. individual checksum). The transmitter will cycle through the RF channels in ascending order, transmitting one packet on each channel. This cycle will be repeated `CYCLE_COUNT` times. Packets are transmitted at a 2 millisecond interval. The total duration of the test (at the receiver) is `0.002 * CYCLE_COUNT * RF_CHANNEL_COUNT` seconds (e.g. 6 minutes for 10000 cycles over 18 channels)		
SPI Operations (pars)	`CMD_REQ(0x13, 8, sw)` **[Byte][bit] index**	**Field**	**Description**
	`[0:1]`	`CYCLE_COUNT`	Number of times to cycle through the RF channel space (0 = infinite). This number should be higher than the corresponding number for the receiver, since some packets are lost while the receiver is synchronizing.
	`[2:5]`	`NWK_ID`	"Network ID" to be used by the PER test. Select either the device ID of the transmitter/receiver used in the test or a random number between 0x01000000 and 0xFFFFFFFD (with a small risk of disturbing or being disturbed by other networks).
	`[6]`	`TX_POW`	Target TX power in dBm (2's complement)
	`[7]`	`RF_CHANNEL`	0: Use the RF channel set used by the application (normally channels 1 through 18, as described in section 2.4.3). 1-18: Use only this single RF channel.
Related events	-		
Usage limitations	Only available in production test operation in FW1.0.5 and later. Is fixed at 5 Mbps RF data rate, even when the application used to generate production test image uses 2 Mbps.		
Notes	A device reset should be performed before and after using this command.		

Command	RFT_RXPER

Description	Runs the receiver side of the packet error rate test. The test uses a packet format similar to the CC85xx radio protocol, with pseudo-random payload data. Each packet contains a header of 16 bytes (incl. individual checksum) and 16 data blocks (corresponding to PCM16 audio slices) of 36 bytes each (incl. individual checksum).
	
	When started, the receiver will attempt to synchronize with the transmitter (running RFT_TXPER), within the specified timeout. Once synchronized, the receiver will iterate through the specified `CYCLE_COUNT`, and then output the result.
	
	Packets are transmitted at a 2 millisecond interval. The total duration of the test (at the receiver) is `0.002 * CYCLE_COUNT * RF_CHANNEL_COUNT` seconds (e.g. 6 minutes for 10000 cycles over 18 channels)
	
	The relationship between the output figures is as follows
	• Number of packets with OK header: `CYCLE_COUNT – (SYNC_ERR_COUNT + HEADER_ERR_COUNT)`
	• Number of packets with one or more slice errors: `CYCLE_COUNT – (ALL_OK_COUNT + SYNC_ERR_COUNT + HEADER_ERR_COUNT)`
	• For unused RF channels: `XXXXX_COUNT = 0` (where `XXXXX` is `ALL_OK`, `SYNC_ERR`, `HEADER_ERR` or `SLICE_ERR`)
	• Sum of all `XXXXX_COUNT = XXXXX_TOTAL` (where `XXXXX` is `ALL_OK`, `SYNC_ERR`, `HEADER_ERR` or `SLICE_ERR`)
	
	The packet error rate at a given channel can be calculated as follows:
	`PER = (CYCLE_COUNT – ALL_OK_COUNT) / CYCLE_COUNT`
	
	The slice error rate at a given channel can be calculated as follows:
	`SER = (16 * (SYNC_ERR_COUNT + HEADER_ERR_COUNT) + SLICE_ERR_COUNT) / (16 * CYCLE_COUNT)`

SPI Operations	`CMD_REQ(0x12, 11, sw)`

(pars)

[Byte][bit] index	Field	Description
[0:1]	CYCLE_COUNT	Number of times to cycle through the RF channel space (0 = invalid).
[2:5]	NWK_ID	"Network ID" to be used by the PER test, which should match the corresponding value on the transmitter.
[6:9]	TIMEOUT	Timeout for synchronizing with the transmitter at startup, in increments of 10 ms.
[10]	RF_CHANNEL	0: Use the RF channel set used by the application (normally channels 1 through 18, as described in section 2.4.3). 1-18: Use only this single RF channel.

`READ(sw, 244, data)`

(data)

[Byte][bit] index	Field	Description
[0:3]	CYCLE_COUNT	Actual number of RF channel cycles.
[4:7]	RF_CHANNEL_COUNT	Number of RF channels used.
[8:11]	ALL_OK_TOTAL	Sum of `ALL_OK_COUNT`.
[12:15]	SYNC_ERR_TOTAL	Sum of `SYNC_ERR_COUNT`.
[16:19]	HEADER_ERR_TOTAL	Sum of `HEADER_ERR_COUNT`.
[20:23]	SLICE_ERR_TOTAL	Sum of `SLICE_ERR_COUNT`.
	For each RF channel *i*=[0,19]	
[24+11*i:25+11*i]	ALL_OK_COUNT	Number of packets received with no errors at all on RF channel *i*.
[26+11*i:27+11*i]	SYNC_ERR_COUNT	Number of packet timeouts on RF channel *i* (no valid synchronization word received).
[28+11*i:29+11*i]	HEADER_ERR_COUNT	Number of packets with header error on RF channel *i*.
[30+11*i:32+11*i]	SLICE_ERR_COUNT	Total number of slice errors in packets with OK header on RF channel *i* (16 slices per packet).
[33+11*i]	RSSI_MEAN	Mean RSSI for packets with OK header on RF channel *i*.
[34+11*i]	RSSI_STD_DEV	Standard deviation of RSSI values for packets with OK header on RF channel *i*.

Related events	-
Usage limitations	Only available in production test operation in FW1.0.5 and later. Is fixed at 5 Mbps RF data rate, even when the application used to generate production test image uses 2 Mbps.
Notes	A device reset should be performed before and after using this command. The packet error rate test should not be used to evaluate audio streaming robustness, which is more sensitive to bursts of errors than the averaged packet error rate. Audio streaming performance should be evaluated using the RF and audio statistics commands, while streaming audio.

Command	RFT_NWKSIM		
Description	Simulates the RF behavior of a protocol master or protocol slave, without need to establish a network connection. This can be used to test regulatory compliance of the radiated power of a single CC85xx device.		
SPI Operations	`CMD_REQ(0x27, 14, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0:1]`	`TIMESLOT_PERIOD`	Timeslot period in us (e.g. 2500)
	`[2:3]`	`TX_MIN_DURATION`	Minimum length of transmitted packet in us (e.g. 600)
	`[4:5]`	`TX_MAX_DURATION`	Maximum length of transmitted packet in us (e.g. 1600)
	`[6:7]`	`RX_DURATION`	Duration of one RX period in us (e.g. 250)
	`[8]`	`SP_COUNT`	Number of protocol slaves to enable RX for. For protocol master: Use 1-4, resulting in 1 TX period + SP_COUNT RX periods per timeslot For protocol slave: Use 0, resulting in 1 RX period + 1 TX period per timeslot
	`[9]`	`TX_POW`	Target TX power in dBm (2's complement)
	`[10:13]`	`ACTIVE_RF_CH_LIST[4]`	Active channels. Must be sorted from low in most significant byte ([10]) to high in the least significant byte ([13]) (e.g. 2, 5, 11, 14).
Related events	-		
Usage limitations	Only available in production test operation. Is fixed at 5 Mbps RF data rate, even when the application used to generate production test image uses 2 Mbps.		
Notes	A device reset should be performed before and after using this command. The parameters should be checked and updated as necessary when switching between different versions of the PurePath Wireless Configurator.		

B.3.12 Audio Test Commands

Command	AT_GEN_TONE		
Description	Enables/disables tone generation on the specified audio channel. Different tones can be generated simultaneously on different audio channels, by executing one AT_GEN_TONE command per channel.		
SPI Operations	`CMD_REQ(0x20, 5, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	`[0]`	`CHANNEL`	Channel index (0 to X-1 for the X channels streamed from the protocol master to protocol slave, if any. 10 to 10+Y-1 for the Y channels streamed from the protocol slave to the protocol master, if any)
	`[1]`	–	Reserved, write 0
	`[2]`	`AMP_TONE`	16-bit sample amplitude value divided by 256, allowed values are 0 and 1 through 128 (for silence or an amplitude ranging from 2^8 to 2^{15}, respectively).
	`[3:4]`	`FREQ_TONE`	Frequency, in increments of 10 Hz, allowed values are 10 to 2000 (for a range from 100 Hz to 20 kHz)
Related events	–		
Usage limitations	Only available in production test operation for non-USB devices. Only valid for audio channels that are configured (in the Configurator) as output on the local serial audio interface.		
Notes	The tone is generated with minimum 60dB SNR at full scale. The command is ignored if the `CHANNEL` parameter is invalid. At power-up/after reset there is a configuration dependent delay before the AT_GEN_TONE command can be used.		

Command	AT_DET_TONE
Description	Estimates amplitude and frequency on the specified audio channel. The command can accurately estimate tones ranging from 250 Hz to 18kHz with 16-bit sample amplitude values from 2^6 to 2^{15}. Results outside these ranges can deviate significantly from expected results.
SPI Operations	`CMD_REQ(0x21, 1, pars, sw)`

(pars)

[Byte][bit] index	Field	Description
[0]	CHANNEL	Channel index (0 to X-1 for the X channels streamed from the protocol master to protocol slave, if any. 10 to 10+Y-1 for the Y channels streamed from the protocol slave to the protocol master, if any)

(data)

`READ(4, sw, data)`

[Byte][bit] index	Field	Description
[0:1]	AMP_TONE	Estimated amplitude
[2:3]	FREQ_TONE	Estimated frequency, in increments of 10 Hz

Related events	-
Usage limitations	Only available in production test operation for non-USB devices. Only valid for audio channels that are configured (in the Configurator) as input on the local serial audio interface.
Notes	The tone detector needs ~10ms (at 44.1kHz) to estimate amplitude and frequency. Executing this command before sufficient data has been collected will return random data. Any value may be returned in AMP_TONE and FREQ_TONE if the CHANNEL parameter is invalid. At power-up/after reset there is a configuration dependent delay before the AT_DET_TONE command can be used.

B.3.13 IO Test Commands

Command	IOTST_INPUT
Description	Selected pins are configured as input and their logical value is returned.
SPI Operations	`CMD_REQ(0x22, 4, pars, sw)`

(pars)

[Byte][bit] index	Field	Description
[0]	–	Reserved, write 0
[1][7:1]	–	Reserved, write 0
[1][0]	XANT_USB_SEL	Read XANTP/USBP and XANTN/USBN if set
[2][7]	–	Reserved, write 0
[2][6]	GIO15_SEL	Read GIO15 if set
[2][5]	GIO14_SEL	Read GIO14 if set
		…
[2][0]	GIO9_SEL	Read GIO9 if set
[3][7]	GIO8_SEL	Read GIO8 if set
		…
[3][0]	GIO1_SEL	Read GIO1 if set

`READ(4, sw, data)`

(data)

[Byte][bit] index	Field	Description
[0]	–	Reserved, ignore value
[1][7:3]	–	Reserved, ignore value
[1][2]	USB_DIFF_VAL	Differential value of XANTP/USBP and XANTN/USBN
[1][1]	XANTP_USBP_VAL	Value of XANTP/USBP
[1][0]	XANTN_USBN_VAL	Value of XANTN/USBN
[2][7]	–	Reserved, ignore value
[2][6]	GIO15_VAL	Value of GIO15
[2][5]	GIO14_VAL	Value of GIO14
		…
[2][0]	GIO9_VAL	Value of GIO9
[3][7]	GIO8_VAL	Value of GIO8
		…
[3][0]	GIO1_VAL	Value of GIO1

Related events	-
Usage limitations	Only available in production test operation.

Command	IOTST_INPUT
Notes	The pins will revert to their original states when their respective <PIN>_SEL is released. Overriding pins for audio clock input to CC85xx (BCLK and WCLK from external source) is not recommended.

Command	IOTST_OUTPUT		
Description	Selected pins are configured as output and driven to the logical value in <PIN>_VAL		
SPI Operations	`CMD_REQ(0x23, 8, pars, sw)`		
(pars)	**[Byte][bit] index**	**Field**	**Description**
	[0]	–	Reserved, write 0
	[1][7:2]	–	Reserved, write 0
	[1][1]	XANT_USB_DIFF_SEL	Write XANTP/USBP and XANTN/USBN if set, using XANT_USB_DIFF_VAL
	[1][0]	XANT_USB_PN_SEL	Write XANTP/USBP and XANTN/USBN if set, using XANT_USB_P_VAL and XANT_USB_N_VAL (don't care if USB_DIFF_SEL is also set)
	[2][7]	–	Reserved, write 0
	[2][6]	GIO15_SEL	Write GIO15 if set
	[2][5]	GIO14_SEL	Write GIO14 if set
			…
	[2][0]	GIO9_SEL	Write GIO9 if set
	[3][7]	GIO8_SEL	Write GIO8 if set
			…
	[3][0]	GIO1_SEL	Write GIO1 if set
	[4]	–	Reserved, write 0
	[5][7:3]	–	Reserved, write 0
	[5][2]	XANT_USB_DIFF_VAL	When using XANT_USB_DIFF_SEL: Differential value of XANTP/USBP and XANTN/USBN
	[5][1]	XANT_USB_P_VAL	When using XANT_USB_PN_SEL: Value of XANTP/USBP
	[5][0]	XANT_USB_N_VAL	When using XANT_USB_PN_SEL: Value of XANTN/USBN
	[6][7]	–	Reserved, write 0
	[6][6]	GIO15_VAL	Value of GIO15
	[6][5]	GIO14_VAL	Value of GIO14
			…
	[6][0]	GIO9_VAL	Value of GIO9
	[7][7]	GIO8_VAL	Value of GIO8
			…
	[7][0]	GIO1_VAL	Value of GIO1
Related events	-		
Usage limitations	Only available in production test operation.		
Notes	The pins will revert to their original states when their respective <PIN>_SEL is released.		

C PCMLF Characteristics

Table 27 - PCMLF filter characteristics

PCMLF filter characteristics (@48Khz)						
	Decimation		Interpolation		Combined	
	Freq. (Hz)	Atten. (dB)	Freq. (Hz)	Atten. (dB)	Freq. (Hz)	Atten. (dB)
Cutoff (@-3dB)	633	-3	5,955	-3	633	-3
Passband (0-f_c)	0-633	[-3;0]	0 – 5,955	[-3;0]	0 - 633	[-3;0]
Transitionband (f_c-f_s)	633-5,850	[-90;-3]	5,955 – 8,850	[-90;-3]	633 – 5,850	[-90;-3]
Stopband (f_s)	5,850-20k	< -90	8,850 – 20k	< -90	5,814 – 20k	< -90
	Filter latency					
Latency (#samples)	≈ 37					
Latency (µs)	≈ 770					

The filters for the PCMLF format are designed for a sample rate of 48 kHz. In a digital system everything scales with sample rate, so for 44.1 kHz system, the upper limit for a full scale signal, which isn't going to create mirror components in the audio range (20Hz-20kHz) is

$$f_u = \frac{20kHz}{48kHz} \cdot fs = 0.417 \cdot 44.1kHz = 18.375kHz$$

All other frequencies defined in

Table 27 scale in a similar way when adjusting the sample rate.

D SLAC Performance Analysis

The table below lists a set of performance estimates for SLAC. The estimates are done using PQevalAudio (http://www-mmsp.ece.mcgill.ca/Documents/Software/Packages/AFsp/PQevalAudio.html) which is a free tool for evaluating perceptual audio degradation of an audio sample. The tool is based on the ITU-R BS.1387 (PEAQ) standard. The tool requires and audio sample to evaluate and a reference sample. The tool outputs a single number defined as the Objective Difference Grade. It's a scale which has the following meaning:

- 0 Imperceptible
- -1 Perceptible but not annoying
- -2 Slightly annoying
- -3 Annoying
- -4 Very annoying

The audio samples used are

- Under The Water - Marie Frank, Ancient Pleasures (Pop/Rock)
- Bolero - Ravel/Bizet and Symphonieorchester Ljubljana, ZYX Classic (Classical)
- Tears In Heaven - Eric Clapton and Will Jennings, Unplugged (Rock)

To put some perspective on the performance numbers, similar evaluations has been done using AAC and MP3 encoded samples. Keep in mind when comparing numbers, that AAC and MP3 are offline encoders. This allows for much more complex encoding techniques. The AAC/MP3 encoded samples have been ripped using standard settings in iTunes v9. The decoding has been done using online available LAME and FAAD decoders.

Table 28 - SLAC performance estimates

PQEval of audio files (two channels @48Khz)							
Artist-Song	**Type**	**Ref**[5]	**SLAC (464kbps)**	**MP3 (128kbps)**	**MP3 (256kbps)**	**AAC (128kbps)**	**AAC (256kbps)**
Marie Frank – Under the Water	Pop/ Electronic	0.215	**-0.066**	-0.945	0.004	-1.007	0.096
Razel/Bizet – Bolero	Classical	0.210	**0.029**	-0.945	-0.072	-1.025	-0.017
Eric Clapton – Tears in Heaven	Rock	0.215	**-0.012**	-0.969	-0.116	-0.494	-0.05

[5] The reference number is the original song compared to itself. This also explains why some of the numbers in the table are positive.

E Revision History

This is the revision history for the CC85xx Family User's Guide document. Refer to the PurePath Wireless Configurator's "Change Log" for the firmware/Configurator revision history.

SWRU250M - Together with FW Release 1.4.2 (June, 2013)

- Updated references to FW release number
- Updated information on RFT_RXTST_RSSI EHIF command usage (section B.3.11)

SWRU250L - Together with FW Release 1.4.1 (November, 2012)

- Updated introduction with overview of new features (section 1.1.3):
  - Left/right balance buttons (for local volume control on autonomous protocol slaves)
  - Local input/output per-channel volume offset in host controlled operation
  - Non-volatile storage made optional on per-value basis in autonomous operation
  - User-specified remote control information, allowing for custom-defined USB HID reports or transfer of proprietary remote control input to a host-controlled protocol master.
- New application supported by this release: Remote controls with audio capabilities (section 1.2.2)
- Updated size of non-volatile storage (with NVS_SET_DATA and NVS_GET_DATA) from two to four 32-bit values (sections 1.3.3.1 and B.3.10)
- Updated overview of PurePath Wireless Configurator settings (section 1.4.1)
- Non-volatile storage in autonomous operation can be disabled on per stored value (multiple sections)
- Added support for custom-defined USB HID reports (sections 1.3.3, 2.1.8.2, 2.2.1.4, 2.5.3 and 3.6.3)
- Added support for custom-defined / proprietary remote control input (sections 1.2.2, 1.3.3, 2.2.1.4, 3.3.3)
- Fixed documentation error: Sample rate selection is not possible through EHIF (section 2.3.2.2)
- Fixed documentation error: For PCMLF slice production, low-pass filtering is done before decimation, not after (Figure 30)
- Changed selection of initial antenna in the antenna diversity algorithm (section 2.4.5)
- Added support for input volume increment/decrement and left/right output balance buttons (sections 3.3.1, 3.5.4, 3.5.4.1)
- Remote control buttons can share GIO pins by using CLICK and HOLD button events instead of STATE (3.3.1)
- Clarified routing of volume control information for remote volume control (section 3.5.4.1)
- Extended VC_SET_VOLUME/VC_GET_VOLUME to support local input/output logical channel offsets at both protocol master and slave (section 3.5.4.3, B.3.7)
- Clarified where smooth volume changes take place (section 3.5.4.4)
- Fixed documentation error: The DI_GET_DEVICE_INFO EHIF command is not available in bootloader mode. The DI_GET_CHIP_INFO EHIF command is available (section B.2.3).
- Added option to let protocol master enter the LOCAL_STANDBY power state if no protocol slaves have been connected for a certain period of time (section 3.7.1.3)
- EHIF command set changes (Table 26):
  - RC_SET_DATA and RC_GET_DATA updated (section B.3.4)
  - NVS_GET_DATA and NVS_SET_DATA expanded (section B.3.10)
  - VC_SET_VOLUME and VC_GET_VOLUME expanded (section B.3.7)

- Several minor clarifications and general typographical cleanups

SWRU250K - Together with FW Release 1.4.0 (June, 2012)

- Updated introduction with overview of new features (sections 1.1.1, 1.1.3, 1.3.1 and 1.3.3). This includes:
  - Input and output silence detection
  - Separate blinking pattern / LED low battery status indication
  - Additional HID support: two-axis mouse with up to 8 buttons
  - Manual frequency planning
  - Support for lower latency settings (down to 512 samples for some configurations)
  - Timeslot alignment
- Multi-master systems now supported with timeslot alignment (section 1.2.2)
- Updated overview of PurePath Wireless Configurator settings (section 1.4.1)
- Added support for 2 Mbps RF raw data rate (section 2.1.1.4)
- Added EP4 to the list of used USB endpoints and clarified EP3 usage (sections 2.1.8.2 and 3.6.3)
- Added support for two-axis mouse with buttons (sections 2.2.1.4, 2.5.3 and 3.3.3)
- Updated new lower latency with table listing limitations (section 2.3.2.3)
- Added possibility to disable soft fade-out for improved audio streaming robustness (section 2.3.3.5)
- Added new section on timeslot alignment (section 2.4.6)
- Added manual frequency planning (sections 2.4.3, 2.4.3.1 and 3.4.5)
- Clarified human user interface description (section 3.3)
- Added new section on mouse movement handling (section 3.3.3.1)
- Updated power management to include analog input silence detection (sections 3.7.1, 3.7.1.3 and 3.7.1.4)
- Added optional GIO pin to indicate the OFF state for host-controlled operation (section 3.7.1.4)
- EHIF command set changes (Table 26):
  - New NWM_SET_RF_CH_MASK command (section B.3.3)
  - RC_SET_DATA and RC_GET_DATA now support mouse movement/buttons (section B.3.4).
- Added new IN_SILENCE_TIME field to the PM_GET_DATA command (section B.3.6)
- Clarified parameters in the PS_AUDIO_STATS and PS_RF_STATS commands (section B.3.8)
- Noted that RFT_TXPER, RFT_RXPER and RFT_NWKSIM only support 5 Mbps RF data rate (section B.3.11)
- Clarified RFT_NWKSIM usage (section B.3.11)
- Updated Figure 5 and Figure 35
- Several minor clarifications and general typographical cleanups

SWRU250J - Together with FW Release 1.3.0 (December, 2011)

- Updated introduction with overview of new features (sections 1.1.1, 1.1.3, 1.3.1, 1.3.2 and 1.3.3). This includes:
  - Proximity-based pairing, allowing for button-less protocol master

  - Advanced power management with battery voltage monitoring, network standby mode and power reduction for the external audio device.
  - Support for microphones with automatic dynamic channel selection
  - Antenna diversity
  - Continuous audio clock generation at protocol slaves to be able to drive external circuitry while not connected to a network
  - USB audio device now supports 24-bit audio streaming in certain configurations
- Removed overviews of planned releases, since FW 1.3.0 completes the originally planned feature set (several sections). New features will be based on customer input
- Wireless microphone applications are now supported (section 1.2.2)
- Updated overview of PurePath Wireless Configurator settings (section 1.4.1)
- Added note on GIO0 not suitable for general purpose I/O usage (section 2.1.2)
- Stated that GIO14 and GIO15 can now be used for other purposes than range extender control (section 2.1.6.1)
- Updated description of battery voltage monitoring interface (section 2.1.7.1)
- Added information on version compatibility to the antenna diversity control interface description (section 2.1.9)
- Improved description of network topology limitations and trade-offs (section 2.2.1)
- Updated information on dynamic channel selection to match new extended options (section 2.3.1)
- SLAC encoding now supports 24-bit input (Table 12 and section 2.3.3.3)
- Added description of antenna diversity algorithm (section 2.4.5)
- Updated entire sections on power management to match new feature set (sections 2.6 and 3.7.1)
- If use of a separate production test image is undesired, a simplified application image production test is now available (section 3.1.4)
- Added network standby toggle button (section 3.3.1)
- Added new options for extended indication options for the network status LED (section 3.3.2.1)
- Added network standby remote control commands, and added version compatibility overview (section 3.3.3)
- Described option for proximity-based pairing, updated the recommended procedure for pairing in host-controlled operation (sections 3.4.2, 3.4.2.1 and 3.4.2.2)
- Completed Table 18 with all possible LED patterns, including the new LOW BATTERY pattern.
- Updated description of dynamic channel selection (section 3.5.1.2)
- Continuous audio clock generation at protocol slaves to be able to drive external circuitry while not connected to a network (section 3.5.2)
- Introduced new external audio device state, SR-SWITCH (section 3.5.3.1), CC85xx-generated audio clock will now always run in the INACTIVE state
- Updated meaning of the EHIF status word's PWR_STATE field (Table 25)
- EHIF command set changes (Table 26):
  - Extra filtering criteria on protocol master pairing signal and RSSI have been added to the NWM_DO_SCAN command to ease implementation of button- and proximity-based pairing in host-controlled operation (section B.3.2). **NB:** Behavior changed from FW 1.2.2
  - New PM_GET_DATA command (section B.3.6)
  - New PM_SET_STATE functionality (B.3.6)
  - New RFT_RXTST_CONT command, renamed the old RFT_RXTST command to RFT_RXTST_RSSI (section B.3.11)

  - New IO test functionality for antenna diversity control pins (B.3.13)
- New field WPM_PM_STATE in NWM_DO_SCAN and NWM_GET_STATUS commands (previously reserved)
- Clarified use of the CHANNEL and AMP_TONE fields of the AT_GEN_TONE and AT_DET_TONE commands (section B.3.12)
- Improved description of the CAL_SET_DATA command (section B.3.9)
- Several minor clarifications and general typographic cleanups

SWRU250I - Together with FW Release 1.2.2 (September, 2011)

- Documented added support for the USB HID class SET_IDLE and GET_IDLE requests (Table 23)
  - USB HID report transmission can now also be triggered by the specified report idle duration (section 3.6.3)
- Updated documentation of USB audio device class GET_RES request (Table 23). It does not support sample rate resolution.

SWRU250H - Together with FW Release 1.2.1 (August, 2011)

- Added documentation of new PCME24 streaming format (several sections)
- Added guidelines/limitations for selecting audio streaming formats (section 2.3.3)
- Stated that USB vendor and product IDs may not be selected at random (section 2.5.1)
- Fixed incorrect command code for RC_GET_DATA (section B.3.4)

SWRU250G - Together with FW Release 1.2.0 (July, 2011)

- Added and updated USB documentation (several sections)
- Added documentation of new remote control feature (several sections)
- Updated documentation regarding bi-directional audio networks (several sections)
- Updated description for remote volume control, (section 3.5.4.1)

SWRU250F - Together with FW Release 1.1.1 (May, 2011)

- Added documentation of new support for user-defined external audio devices (sections 1.1.3 and 3.5.3). Refer to the PurePath Wireless Configurator's help system for full details.
  - Volume control now allows logical channel offsets to be used by the individual channels on multi-channel protocol slaves. Using this feature extension is optional, and must be enabled at configuration time (sections 3.5.3 and B.3.7).
- Corrected release revision for battery monitor functionality (section 2.1.7)

SWRU250E - Together with FW Release 1.1.0 (April 1, 2011)

- Added documentation of new features
  - "Automatic Protocol Slave power-down" (several sections)
  - Dynamic channel selection cycling using button (several sections)
  - SLAC streaming format. (section 2.3.3.3 and section D)
- Updated Table 1 to match actual firmware releases (section 1.1.2)
- Updated feature set to match to match actual firmware releases (section 1.1.3)
- Updated documentation regarding interoperability (section 1.1.4)
- Clarified rules for protocol slave compatibility (section 1.2.1)

- Updated documentation regarding applications (section 1.2.2)
- Updated documentation of MISO pin when CSn is deasserted (several sections)
- Clarified that FW only supports write-only I^2C operations when customizing audio device drivers (section 2.1.4.2)
- Updated documentation about USB features planned for FW1.2 (section 2.1.8)
- Updated documentation about network topology (section 2.2.1)
- Updated section on audio stream routing to match FW1.1.x (section 2.3.1)
- Updated Table 12 to match FW1.1.x (section 2.3.3)
- Updated unclear timeslot documentation (section 2.4.2)
- Clarified manufacturer ID uniqueness mechanism (section 3.4.1)
- Updated unclear documentation of autonomous operation mode pairing (section 3.4.2.1)
- Updated Table 18 to match FW1.1.x (section 0)
- Clarified data side-channel behavior when multiple protocol slaves are connected
- Updated behavior when writing datagrams to TX Queue and clarified DSC_TX_DATAGRAM and DSC_RX_DATAGRAM documentation. (section 3.4.4.2.1, 3.4.4.2.2, 3.4.4.3 and B.3.4)
- Clarified basic SPI operations figures with regards to need for waiting for MISO to go high. (section B.1)
- Aligned logical channel names in NWM_ACH_SET_USAGE with Configurator naming. (section B.3.3)
- Updated documentation for `CAL_SET_DATA` and `CAL_GET_DATA` (Table 26 and section B.3.9)
- Updated documentation for PS_AUDIO_STATS and PS_RF_STATS (section B.3.8)
- Updated documentation for Audio Test Commands (section B.3.12)
- Updated documentation for IOTST_INPUT (section B.3.13)
- Several minor clarifications and general typographic cleanups.

SWRU250D - Together with FW Release 1.0.5 (Feb 28, 2011)

- Updated Figure 46 to match current behavior, as of FW1.0.3 (section 3.4.2.2)
- Updated Table 24. Size of the DATA+CRC field was wrong. Clarified that fields are big-endian. (section B.2.4)
- Updated Table 26: Wrong command ID and parameter size for RFT_NWKSIM (section B.3).
- NWM_GET_STATUS and NWM_DO_JOIN: The LATENCY field will never be 0, removed text (section B.3.3)
- Clarified DSC_TX_DATAGRAM and DSC_RX_DATAGRAM documentation. The behavior of the commands is unchanged (section B.3.4)
- Updated RFT_TXPER and RFT_RXPER documentation to match new implementation in FW1.0.5 (section B.3.11)
- EHIF command set: Clarified that all reserved fields shall be set to 0 when written and ignored when read (section B.3)
- Corrected the extra latency introduced by PCMLF filters (section C)
- The RFT_TXPER, RFT_RXPER and RFT_NWKSIM EHIF commands are now named consistently

SWRU250C - Together with FW Release 1.0.3 (Nov 30, 2010)

- Added documentation of new EHIF commands NVS_SET_DATA, NVS_GET_DATA, IO_GET_PIN_VAL_DATA

- Updated documentation of the EVT_NWK_CHG bit in several sections.
  NB: Behavior changed from FW1.0.2
- Updated documentation regarding interoperability (section 1.1.4)
- Updated connection recommendations (sections 2.1.8.1 and 2.1.9.1)
- Removed invalid PAIRING state (section 0)
- Added bullet to list of host processor responsibilities (section 3.4.4.3)
- Corrected SYS_RESET and BOOT_RESET documentation (section B.1)
- Removed EHIF command execution time from appendix B, and replaced with general statement in section B.3
- Updated JOIN_TO documentation – there is no infinite timeout (section B.3.3)
- Updated documentation of EHIF RF Test commands (section B.3.11)
- Updated documentation of AT_DET_TONE command (section B.3.12)
- Corrected byte indexes in IOTST_OUTPUT (section B.3.13)

SWRU250B - CC85xx Family User's Guide update only (Sept 17, 2010)

- Updated Table 2 with correct descriptions of XLNAEN and XPAEN.
- Updated Figure 2 and Figure 4 with the TLV320AIC3204 instead of TLV320AIC3101.

SWRU250A – Together with FW Release 1.0.1 (July 16, 2010)

Maintenance release:

- Added a revision history appendix
- Specified the added latency when using PCMLF format (section 2.3.2.3)
- Added documentation of digital input valid pin configuration in Pure Path Wireless Configurator (section 3.5.3)
- Corrected interrupt polarity in EHIF command EHC_EVT_MASK (section B.3.2)
- Updated documentation of EHIF command RFT_RXTST (section B.3.10)
- Updated documentation of EHIF command AT_GEN_TONE (section B.3.12)
- Updated documentation of EHIF command AT_DET_TONE (section B.3.12)

SWRU250 - Together with FW Release 1.0.0 (June 30, 2010)

First public release

F Additional Information and Resources

Texas Instruments offers a wide selection of cost-effective, low-power RF solutions for proprietary and standard-based wireless applications for use in industrial and consumer applications. Our selection includes RF transceivers, RF transmitters, RF front end and System-on-Chips as well as various software solutions for the sub-1 and 2.4 GHz frequency bands. In addition, Texas Instruments provides a large selection of support collateral such as development tools, technical documentation, reference designs, application expertise, customer support, 3rd party and university programs. The Low-power RF E2E Online Community provides you with technical support forums, videos and blogs, and the chance to interact with fellow engineers from all over the world. With a broad selection of product solutions, end application possibilities, and the range of technical support Texas Instruments offers the broadest low-power RF portfolio.

F.1 Texas Instruments Low-Power RF Web Site

Texas Instruments' Low-Power RF Web site has all our latest products, application and design notes, FAQ section, news and events updates, and much more. Just go to http://www.ti.com/lprf.

F.2 Low-Power RF Online Community

- Forums, videos, and blogs
- RF design help
- E2E interaction - Posting own and reading other user's questions

Join us today at http://www.ti.com/lprf-forum

F.3 Texas Instruments Low-Power RF Developer Network

Texas Instruments has launched an extensive network of Low-Power RF development partners to help customers speed up their application development. The network consists of recommended companies, RF consultants, and independent design houses that provide a series of hardware module products and design services, including:

- RF circuit, low-power RF and ZigBee design services
- Low-power RF and ZigBee module solutions and development tools
- RF certification services and RF circuit manufacturing

Need help with modules, engineering services or development tools?

Search the Low-Power RF Developer Network tool to find a suitable partner! http://www.ti.com/lprfnetwork

F.4 Low-Power RF eNewsletter

The Low-Power RF eNewsletter keeps you up to date on new products, news releases, developers' news, and other news and events associated with Low-power RF Products from TI. The Low-Power RF eNewsletter articles include links to get more online information.

Sign up today on http://www.ti.com/lprfnewsletter

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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