TI Designs

SimpleLink™ Wi-Fi® CC3200 Smart Plug Design Guide

TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDC-CC3200SMARTPLUG	Tool Folder Containing Design Files
CC3200	Product Folder
UCC28910	Product Folder
TPS61097A	

TI E2E™ Community

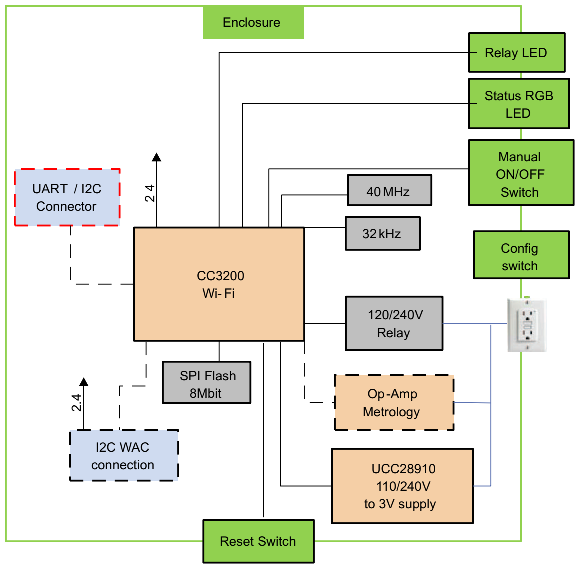
ASK Our E2E Experts
WEBENCH® Calculator Tools

Design Features

- SimpleLink™ Wi-Fi® Connectivity Over 802.11 b/g/n Networks From Any Mobile Device
- Single-Phase Energy Measurement That Calculates Current, Voltage, Power, And Energy
- Single Wireless MCU With Integrated ADC To Compute Metrology Data And Handle Wi-Fi Activities
- Discrete Metrology Circuitry On Board
- Fast-Switching Solid-State Relay To Locally Or Remotely Set The Power On Or Off
- Compact Physical Design With Minimal BOM
- Isolated Flyback Power Supply To Provide Constant-Voltage (CV) and Constant-Current (CC) Output Regulation Without Optical Coupler
- Complete Documentation, Schematic Files, And SW Source For Embedded And Android Mobile Application

Featured Applications

- Smart Plugs Or Power Strips
- Building And Home Monitoring
- Home Appliances And White Goods
- Energy Awareness Related Products



All trademarks are the property of their respective owners.

1 System Description

1.1 Introduction

Increasingly, applications require Wi-Fi-enabled energy measurement and control, particularly in end products such as smart plugs and smart receptacles, and in general application spaces such as home appliances, smart grid, and building automation. This guide offers a form-factor reference design and application software for the CC3200 Smart Plug, while also addressing application requirements in connected applications.

1.2 Energy Measurement and Any Cloud-Connected Device

The Wi-Fi-Enabled CC3200 Smart Plug showcases TI’s ability to provide full system solutions that single competitors cannot offer. Leveraging the portfolio competitiveness and breadth of TI’s wireless connectivity, analog and metrology products enables innovative system features and cost savings.

The CC3200 Smart Plug utilizes discrete components to monitor energy consumption for a single load. This data is then passed to a Wi-Fi device (CC3200) to communicate the data to both another device in the LAN and to a Cloud server. The whole system is powered from a highly compact and efficient primary side regulated flyback PSU using the UCC28910D. A solid-state relay enables the application to control the load based on its energy consumption.

The final design of the CC3200 Smart Plug also includes male and female connections to a NEMA 15-type North American power outlet, making it easy to connect in a series with a device to measure the live power consumption. In addition to the PCB mounted high-voltage connections, a simple enclosure has been designed and provided to insulate any user from potential exposure to high voltages. The electronics are already designed for international voltages, and only the mechanics need to be changed.

1.3 Block Diagram

Figure 1 shows a high-level overview of the connections between the all the various devices in the CC3200 Smart Plug reference design. The only physical connections from the system are the AC voltage input and output for an AC load. On this high-voltage line, the flyback power supply, load control relay, and metrology sensors are connected.

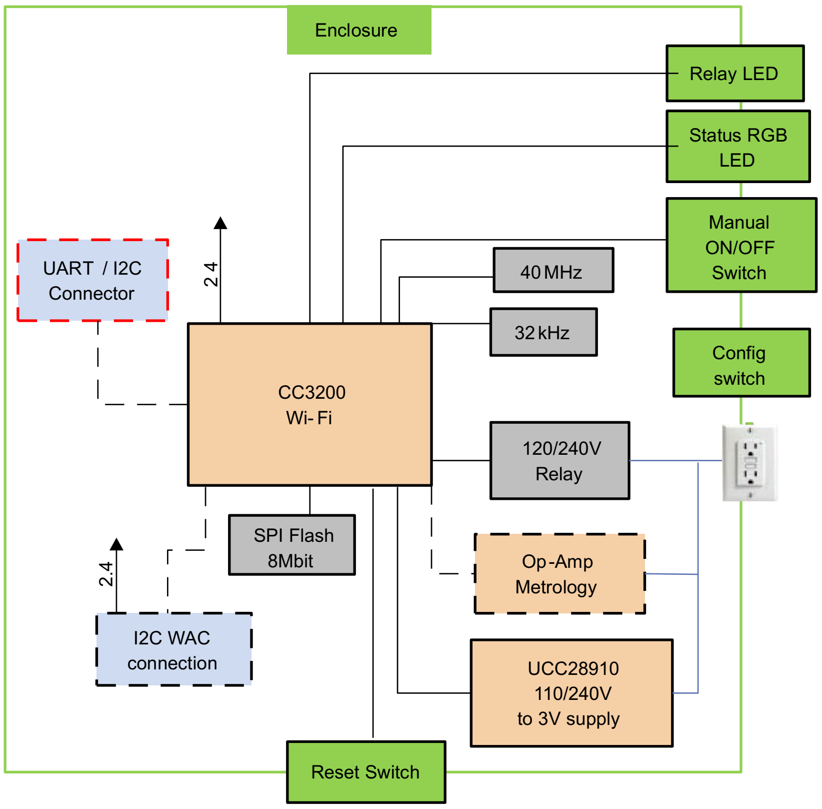


Figure 1. Smart Plug Block Diagram

2 Circuit Design and Component Selection

2.1 Block Diagram

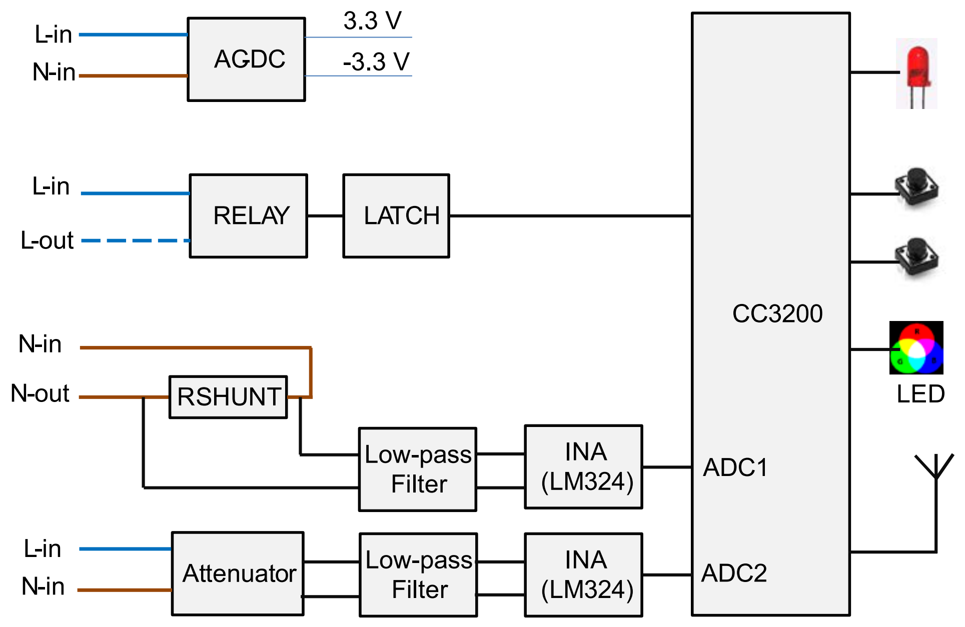


Figure 2. Functional Block Diagram

2.2 Features

- Mechanical relay to switch loads up to 15 A
- 110- / 230-V, 50- / 60-Hz operation
- Very low quiescent current
- Multi-color LED indication for various states
- Voltage, current, frequency, power, and power factor monitoring
- 2.4-GHz Wi-Fi support for remote monitoring and administration
- Cloud support for controlling multiple CC3200_Smart_Plugs from one interface

2.3 Hardware Description

2.3.1 AC-to-DC Converter (Flyback)

The CC3200_Smart_Plug board derives the DC supply for its low-voltage components using a fly-back converter. This converter powers the relay coil and generates a nominal voltage of 3.3 V and -3.3 V needed for the CC3200 and the LM324 IC.

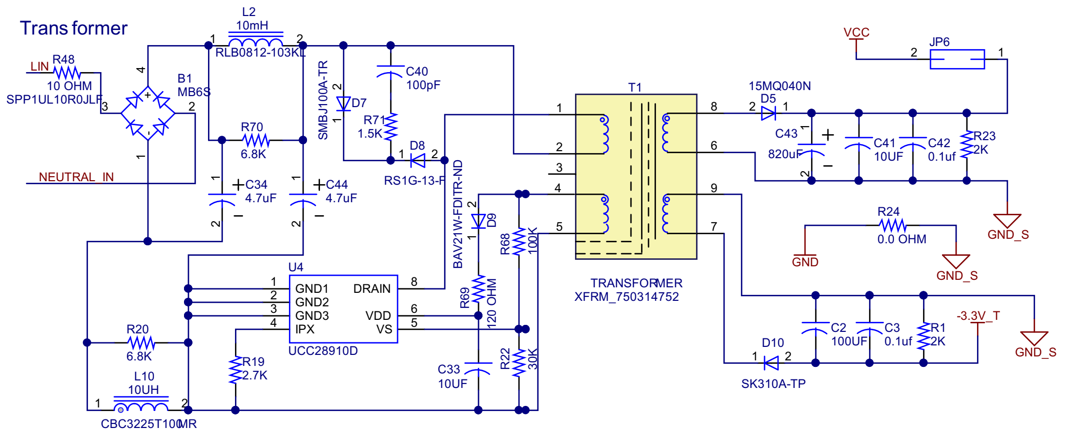


Figure 3. AC-to-DC Conversion

The converter is based on the UCC28910D power MOSFET with integrated voltage sensor. The 110-V / 230-V AC is converted to a corresponding DC voltage using the bridge rectifier B1 and the following filter consisting of C34, L2, and C44. The transformer and the UCC IC switch this DC to generate the required 3.3 V and -3.3 V.

2.3.2 Relay Control

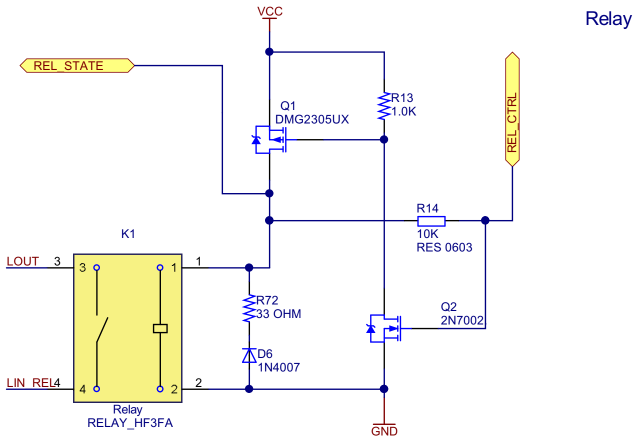


Figure 4. Relay Control Logic

The relay used here is capable of working from 3-V DC, and the coil is driven using a PMOS due to the limited drive capability of the CC3200. To preserve the states of the relay during the CC3200 hibernate or RESET, a latch is formed by using another NMOS transistor. The relay can be turned ON and OFF by high-going or low-going pulses at the gate of transistor Q3. The relay control is driven by a GPIO from the CC3200.

2.3.3 Metrology

The metrology section consists of two parts: one part is the sensing section, which is used to sense the voltage and the current; the other part is a differential- to single-ended converter to drive the ADCs on the CC3200.

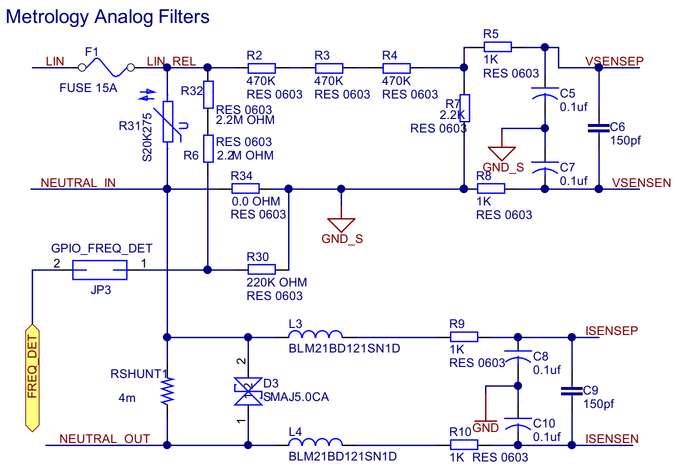


Figure 5. Low Pass Filter

Figure 5 shows the sensing of the voltage and current taken by the load. The input voltage is attenuated using the multiple 470-Kohm and the 2.2-K resistor, which is further filtered using the single-pole RC filter formed by the 1-K, 0.1-µF, and the 150-pF capacitor. Similarly, the current is converted to voltage using the 4-mill-ohm resistor and then filtered using the RC filter and ferrite bead. The D3 limits the peak-to-peak swing of this voltage. The differential voltage is then fed to the LM324-based instrumentation amplifier.

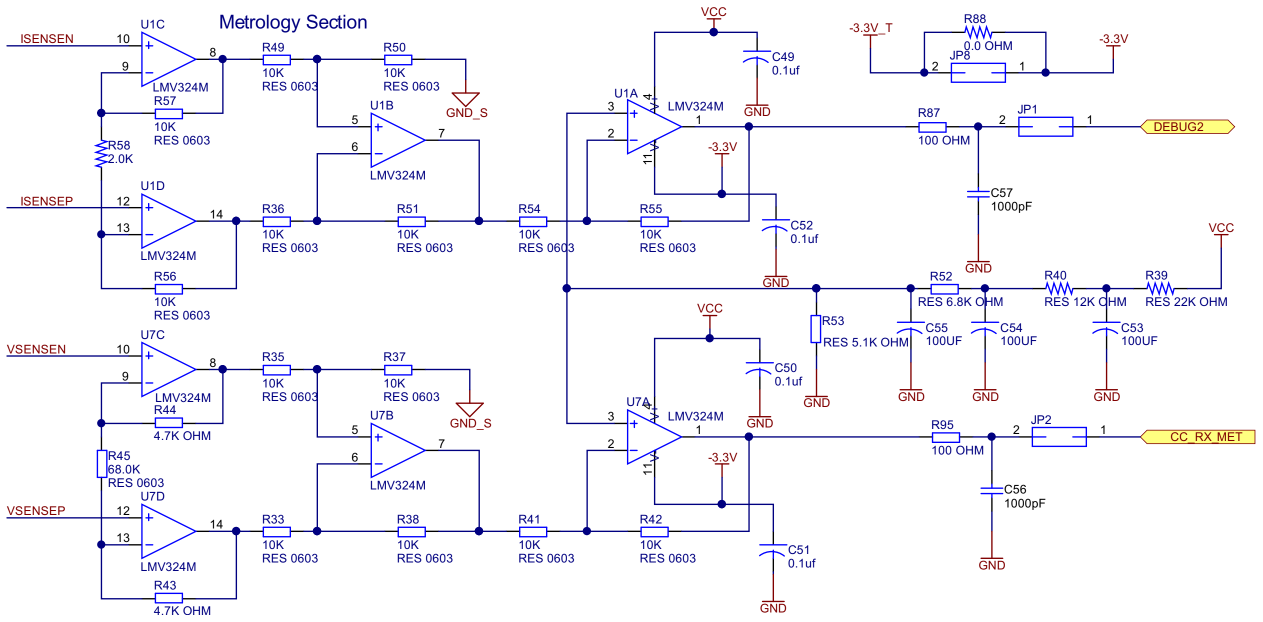


Figure 6. Differential- to Single-Ended Converter

The LM324 is used here as an instrumentation amplifier and also a differential- to single-ended conversion to drive the inputs of the ADC. No output filter exists at the last stage, but a simple RC filter with 100 Ohm and 1µF is recommended to be connected in the next version of the board.

2.3.4 Programming Interfaces

The multiple header pins on the CC3200_Smart_Plug board perform debugs. Some of these are listed below.

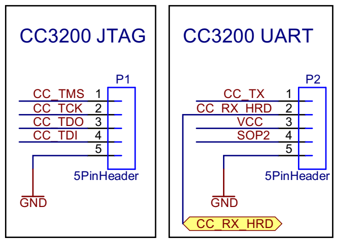


Figure 7. JTAG and UART Pins

The JTAG connector can be used to debug the firmware using an external debugger. The UART header programs the serial flash connected to the CC3200.

2.3.5 Provided Buttons

2.3.5.1 Provisioning Options

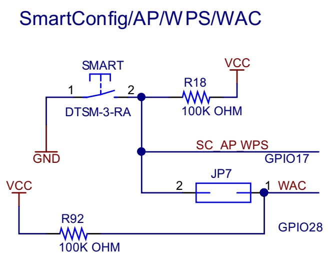


Figure 8. Provisioning Button

The push-button S1 sets the CC3200 device to smartconfig mode. The blue LED will flash at a fast rate when the device enters the smartconfig state.

2.3.5.2 CC3200 Reset Button

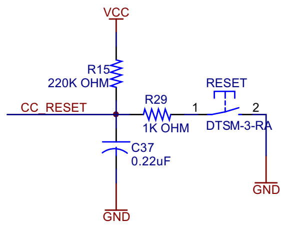


Figure 9. CC3200 Reset Button

The push-button “RESET” resets the CC3200 device at any instance. This action is typically required during the development phase for updating the application code and service packs.

2.3.5.3 Relay Control Button

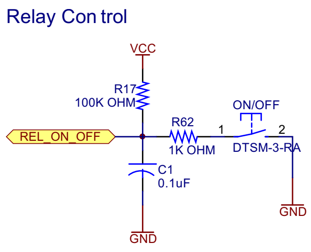


Figure 10. Relay Control Button

The push-button “ON / OFF” toggles the state of the relay, which controls the connected appliance. The signal from the switch is used as an input by the CC3200 to control the relay.

3 Embedded Software System

The software is described in the following subsections. Section 3.1 describes the setup of various peripherals of the MSP430™. Subsequently, the entire metrology software is described as two major processes: the foreground process and background process.

The CC3200_Smart_Plug embedded software mainly consists of two parts: the first one is real time metrology energy computation module; the second one is embedded network software to communicate with secured Cloud and Android mobile applications.

The Metrology software module processes real-time voltage and current ADC channels samples by using advanced signal-processing algorithms. The output parameters of the Metrology software module will be used by CC3200_Smart_Plug to communicate with the user’s cloud or mobile application.

The embedded network software module queries metrology data from the real-time Metrology software module, forms it into packets, and broadcasts the metrology data packets to the companion Android application and Exosite Cloud. The embedded network software module also provides bidirectional communication to increase the user experience with this reference.

3.1 CC3200_Smart_Plug Block Diagram

Figure 11 shows the Metrology and Network modules of CC3200_Smart_Plug.

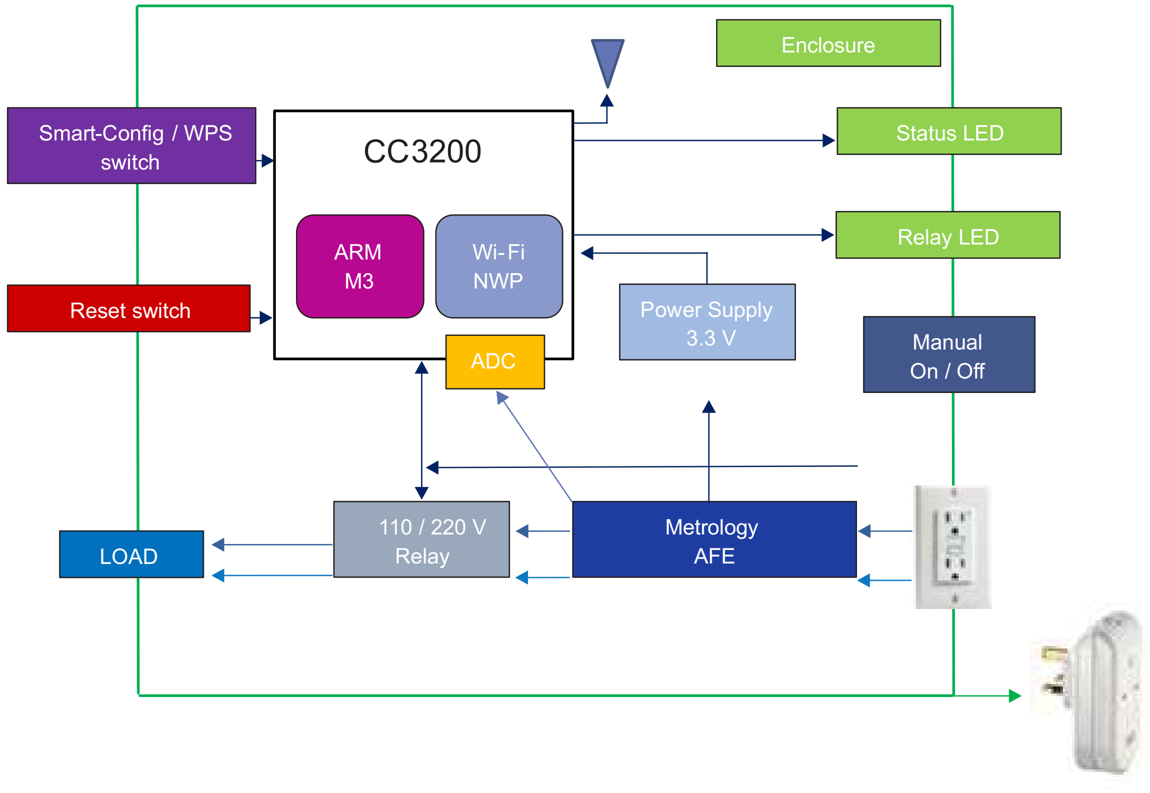


Figure 11. CC3200_Smart_Plug Block Diagram

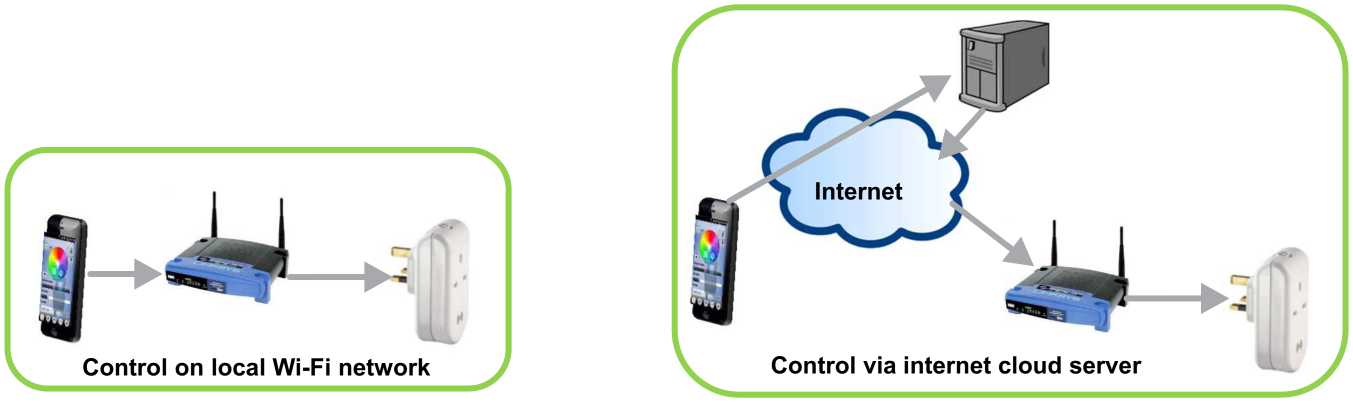

Internet
Control on local Wi-Fi network
Control via internet cloud server

Figure 12. Controls via Local-Network and Cloud-Server Connections

3.2 Metrology Software Module

3.2.1 Assumptions

3.2.1.1 Hardware

- Main supply ground is isolated from Metrology hardware
- The CC3200 ADC input dynamic range is between 0 to 1.46 V
- The voltage and current sense analog input to CC3200 ADC has bias voltage of ~0.72 V
- The voltage and current sense analog signals having dynamic range between 0 to 1.46 V centered at ~0.72 V
- CC3200 supports parallel DMA for ADC channels
- A dedicated GPIO is present to generate a Wh pulse when 1 Wh energy is consumed
- The accuracy and characteristics of ADC channels are identical
- The ADC supports 10 ENOB (Effective Number Of Bits) accuracy
- The low-pass filters at ADC inputs must have same time constants
- The current and voltage channels are 180 degrees out of phase at ADC input

3.2.1.2 Software

- The voltage and current channels scaling factor is determined by accurate calibration by using known load and supply
- The voltage and current channel bias cannot be less than 0.5 V
- The supply AC frequency cannot be out of range between 40 Hz to 70 Hz
- The supply AC voltage cannot be out of range between 10 V to 300 V
- The ADC code word range from 0 to 4095 (12 bits)
- The ADC max voltage is 1.46666 V and corresponds to code word 4095
- The ADC sampling frequency is 62.5 KHz

3.2.2 Software Requirements

This section describes the high-level requirements of the metrology software module, which must compute the parameters shown in Table 1.

Table 1. Metrology Parameters Range and Minimum Resolution

DATA	UNIT	RESOLUTION	RANGE	COMMENT
RMS Voltage	V	0.001	50 V – 300 V	Do not round off any data
RMS Current	A	0.001	0 A – 15 A	Hardware must support a max current sense of 15A
True (Active) Power	W	0.001	0 W – 4500 W	V.I (dot product)
Frequency	Hz	0.001	40 Hz – 70 Hz	50 and 60 Hz mostly used
Apparent Power	VA	0.001	0 VA – 4500 VA	$V_{RMS} \times I_{RMS}$
Reactive P	RVA	0.001	0 RVA – 4500 RVA	
Power Factor	Cos $	0.0001	0 – 1	
Total Energy Consumption	KWh	0.0001	0 – 429496 KWh	The 4 bytes resolution is sufficient to accumulate energy of 4.5 KW load for 11 years
Total Apparent Energy Consumption	KVAh	0.0001	0 – 429496 KWh	
Average True Power	W	0.001	0 W – 4500 W	Average power consumed after CC3200_Smart_Plug ON

3.2.3 Digital Signal Processing Algorithms

This section describes high-level DSP algorithms used to process ADC samples to meet the accuracy of metrology parameters.

3.2.3.1 ADC Data Samples Collection

The CC3200 Supports DMA for ADC to dump samples into memory. Samples must be collected from both channels without any time delay, and the time stamp of each channel sample must match the time stamp of another channel's sample. Ping and Pong buffer approach is recommended for collecting real time samples from ADC using DMA.

3.2.3.2 Improving ADC Accuracy

The CC3200 ADC has 10 ENOB. A high-accuracy quality e-meter needs at least 16 bits of Sigma-Delta ADC (need high linearity) and typical standards require at least 14 bits or more ENOB.

The ADC samples are oversampled by using a 62.5-KHz sampling frequency, considering a frequency of supply voltage that is in the range of 40-70 Hz. These oversamples can improve the ENOB and quantization noise of CC3200 ADC by decimating these samples through block summing.

Every 64 (2^6) samples can be summed and right shifted by 3 bits, which gives 64:1 decimation with three bits improvement in ENOB. This event results in a 13-bit ADC with 0.9765-K samples per second.

This ADC Metrology software module, given the 10 ENOB, can mainly achieve as low as 2% measurement accuracy. This 2% measurement accuracy is acceptable for consumer-based CC3200 Smart Plug applications that do not require a higher level of accuracy.

3.2.3.3 Filtering

The presence of instantaneous high-frequency noise or harmonics in ADC channels must be removed digitally from the channels after sampling by using a basic low-pass filter.

The 5th order FIR filter with 300 Hz cutoff frequency can eliminate high-frequency noise in the channel. The 64:1 decimation also acts as a low-pass filter and helps remove high-frequency noise.

3.2.3.4 Channel DC Offset Calculation

CC3200 ADC dynamic range supports only 0 V – 1.46 V. To fit the current and voltage analog signals in this range, an offset voltage of ~0.72 (mid) has been added to these signals at the input of ADC by using OPAMPs.

If the integration period is not an integral multiple of the period, the DC offset computation is difficult, and it is slightly affected by residuals. The wrong DC offset computation can affect RMS and power-computation values.

The DC estimation must occur using an integral number of the cycle's samples. To know the exact integral number of cycles in an integration period, one must know the frequency of incoming data.

The frequency of incoming data can be computed by using the zero-crossing mechanism. The voltage channel must be used to compute frequency because the current can be zero when no load is connected. The accuracy of frequency can be improved by using residual samples out of full cycles.

3.2.3.5 Metrology Parameters Computation and Output

The metrology parameters must be computed and sent out by integrating samples over every 1 sec. The 1-sec integration drastically reduces the noise in the metrology parameters by high SNR. This section briefly describes the formulas used for the voltage, current, and energy calculations.

Voltage and Current

RMS Voltage (V_{RMS}) and Current (I_{RMS}):

$$V_{RMS} = K_v \bullet \sqrt{\frac{\sum_{n=1}^{\text{Sample count}} V^2(n)}{\text{Sample count}}}$$

$$I_{RMS} = K_i \bullet \sqrt{\frac{\sum_{n=1}^{\text{Sample count}} i^2(n)}{\text{Sample count}}} \quad (1)$$

Where v(n) = Voltage sample at a sample instant 'n' after removing DC offset
I(n) = Current sample at a sample instant 'n' after removing DC offset
Sample count = Number of samples in 1 second
Kv: Scaling factor for voltage. It converts ADC voltage range to supply voltage
Ki: Scaling factor for current. It converts ADC voltage range to supply Current

Power and Energy

Active Power (Pact):

$$P_{ACT} = K_p \times \frac{\sum_{n=1}^{\text{Sample count}} v(n) \times i(n)}{\text{Sample count}} \quad (2)$$

Kp = Scaling factor for power (Kv × Ki)

Apparent Power (Papt):

$$P_{APT} = V_{RMS} \bullet I_{RMS} \quad (3)$$

Reactive Power (Preact):

$$P_{APT}{}^2 = P_{ACT}{}^2 + P_{REACT}2 \quad (4)$$

Active Energy (Eact):

$$E_{ACT} = \sum_{n=1} P_{ACT} \quad (5)$$

Accumulation of power every sec gives energy consumed until the last second.

3.2.4 Metrology Software Module Design and Development

3.2.4.1 Introduction to AFE Hardware

This section briefly explains metrology Analog Front End (AFE) hardware design aspects.

3.2.4.1.1 Vsense and Isense Hardware Logic

Figure 13 shows how hardware logic derives Vsense and Isense from supply and load. The derived analog signal range would be +/-0.7 V with the center at 0 V.

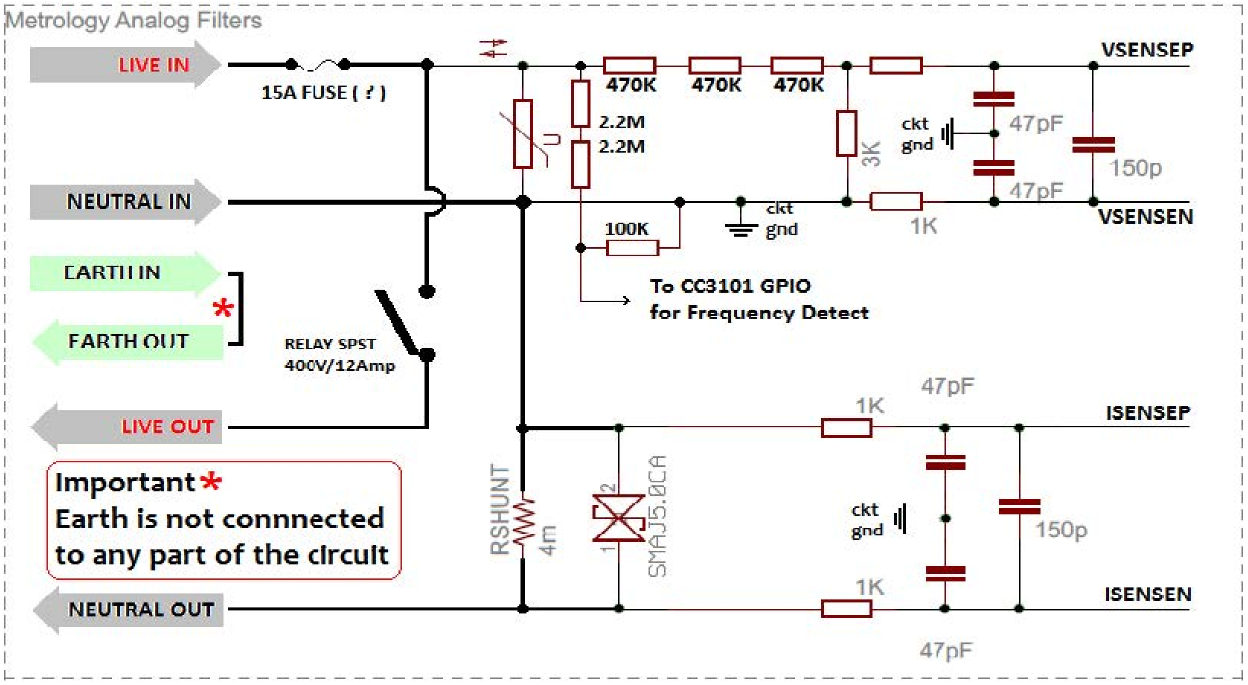


Figure 13. Vsense and Isense Hardware Logic

3.2.4.1.2 Vsense and Isense Dynamic Range Conversion Hardware Logic

Figure 14 shows how hardware logic converts the dynamic range of Vsense and Isense from +/-0.7 V to 0 – 1.45 V with the center at ~+0.72 V.

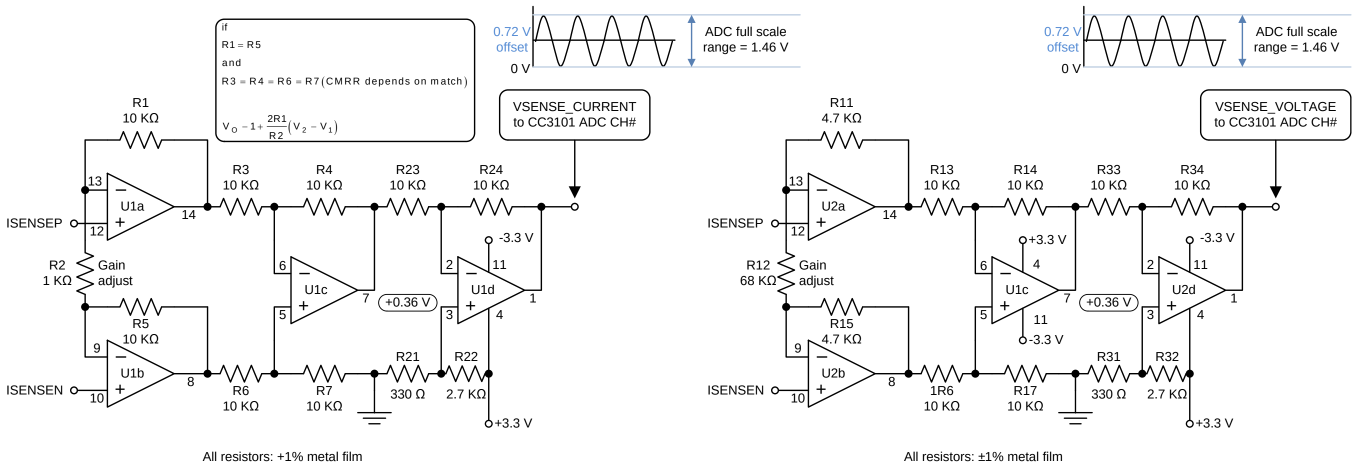


Figure 14. Vsense and Isense Dynamic Range Conversion Hardware Logic

3.2.4.2 Load On / Off Relay Control Logic

The supply load can be cut off or on by using the software GPIO. Figure 15 shows this event in the relay hardware logic.

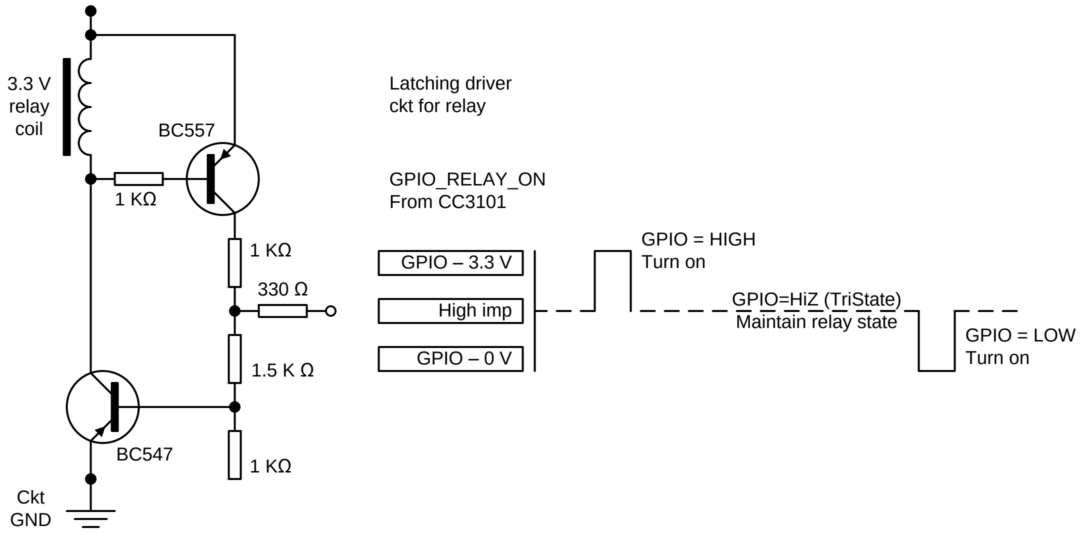


Figure 15. Load On / Off Relay Control Logic

3.2.5 ADC Sample Collection Front End

The analog Vsense and Isense signals from hardware AFE is fed to ADC channels of CC3200 as shown in Figure 16.

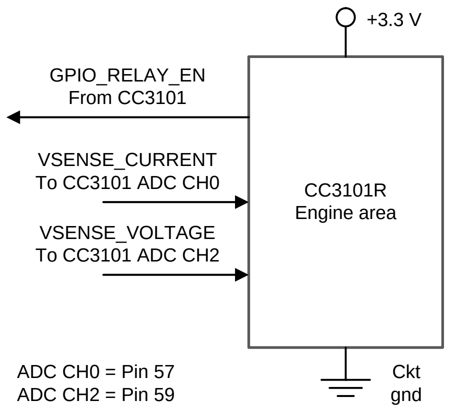


Figure 16. Vsense and Isense Inputs to CC3200

The Input Vsense and Isense analog signal at ADC input as shown in Figure 17 (50-Hz Supply)

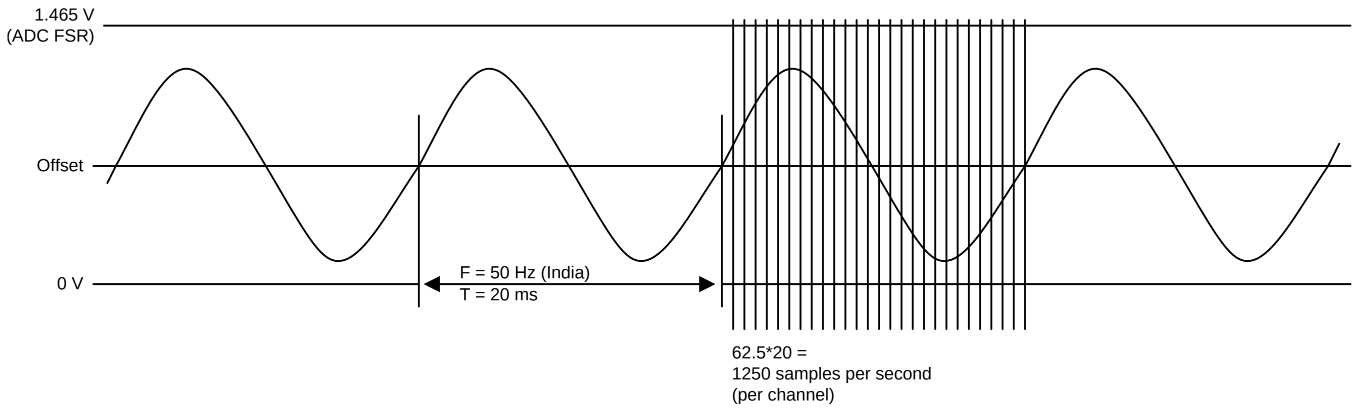


Figure 17. Analog Inputs to the CC3200's ADC

3.2.5.1 *ADC and DMA Configuration*

The CC3200 ADC supports micro-DMA access. DMA is used to collect ADC samples at the rate of 62.5-K samples / sec for signal processing. Figure 18 shows the high-level block diagram of samples collection; the micro DMA can configure to dump a maximum of 1024 samples into the RAM buffer.

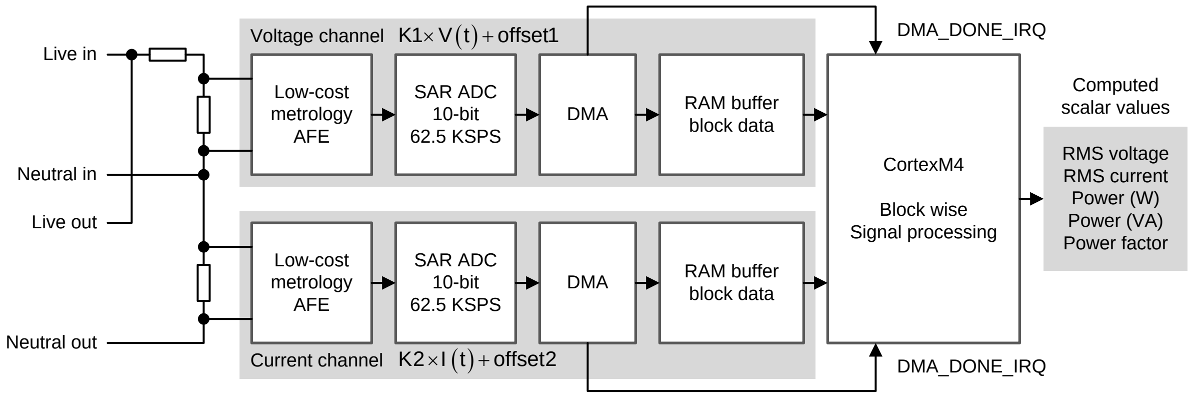


Figure 18. ADC and DMA Configuration

For continuous real-time samples collection, DMA is configured in ping-and-pong buffer transfer mode.

Pins 57 and 59 of CC3200 are muxed to ADC channel 0 and 2 respectively. In software, these two pins are configured as the ADC input.

The ADC channel 0 and corresponding DMA channel 14 collect analog Voltage samples; the ADC channel 2 and corresponding DMA channel 16 are used to collect analog current samples. The sampling frequency is 62.5 KHz, and signal processing algorithm is using 64:1 decimation (explained in detail in next chapter). To keep buffer handling and signal processing algorithm simple, the DMA ping-and-pong buffer collection is selected as a multiple of 64, which is 640 samples (10.24 ms). Every 10.24 ms, once the voltage channel and current's channels DMA interrupts trigger, the M4 collects the 640 samples in the ping buffer, and the DMA continues to collect immediate samples into the pong buffer. The voltage channel interrupt number is 30 and Current channel interrupt number is 32 in ARM M4 processor vector table.

The configured ADC timer enables time stamps in each ADC sample. The raw ADC samples from DMA o / p buffer would be 32 bits having MSB 18-bits time stamp (time stamp counter is 17 bit + 1 dummy bit) and LSB 14 bits for ADC (12-bit ADC + 2LSB dummy bits). The 12-bit ADC samples are extracted from 32-bit samples as 12bit_adc_sample = (32bit_adc_sample >> 2) and 0 × FFF.

In M4 ADC DMA interrupt handler (in both the V-channel and I-channel handler), source of interrupt (ADC DMA channel source) is cleared as soon as interrupt is serviced to service interrupt for next ping/pong DMA done interrupt. The ping-or-pong source of DMA-done interrupt must be identified before processing ADC samples. Re-enable the ping-or-pong source after processing the samples.

3.2.5.2 Metrology Parameters Computation Interval from DMA Interval

With this DMA configuration, two subsequent (V and I) DMA-done back-to-back interrupts trigger M4 to collect and process the data every 10.24 ms. This process eases the processing complexity the metrology parameters compute every 1.024 sec instead of 1 sec, which is 100 samples of 10.24 ms DMA-done interrupts.

In 1.024 sec, a total of 64000 raw ADC samples are used to compute metrology parameters. These oversampled ADC data are good enough to improve the accuracy of the ADC and to reduce quantization noise in the samples.

3.2.6 Digital Signal Processing of ADC samples

Every 10.24 ms, 640 ADC samples (V and I) are available for processing in M4. The processing of these samples are split in to two parts: one part is the 10.24-ms process, which is processing 640 samples to get a good number of decimated samples and accumulating these decimated samples to compute raw metrology parameters; the other part is the 1.024-sec process, which is the final processing of accumulated samples to compute metrology parameters.

3.2.6.1 10.24-ms Process

In the 10.24-ms process, most of the signal processing is done to compute raw metrology parameters. This section describes these signal processing technics one by one. Figure 19 shows the flow chart of this process.

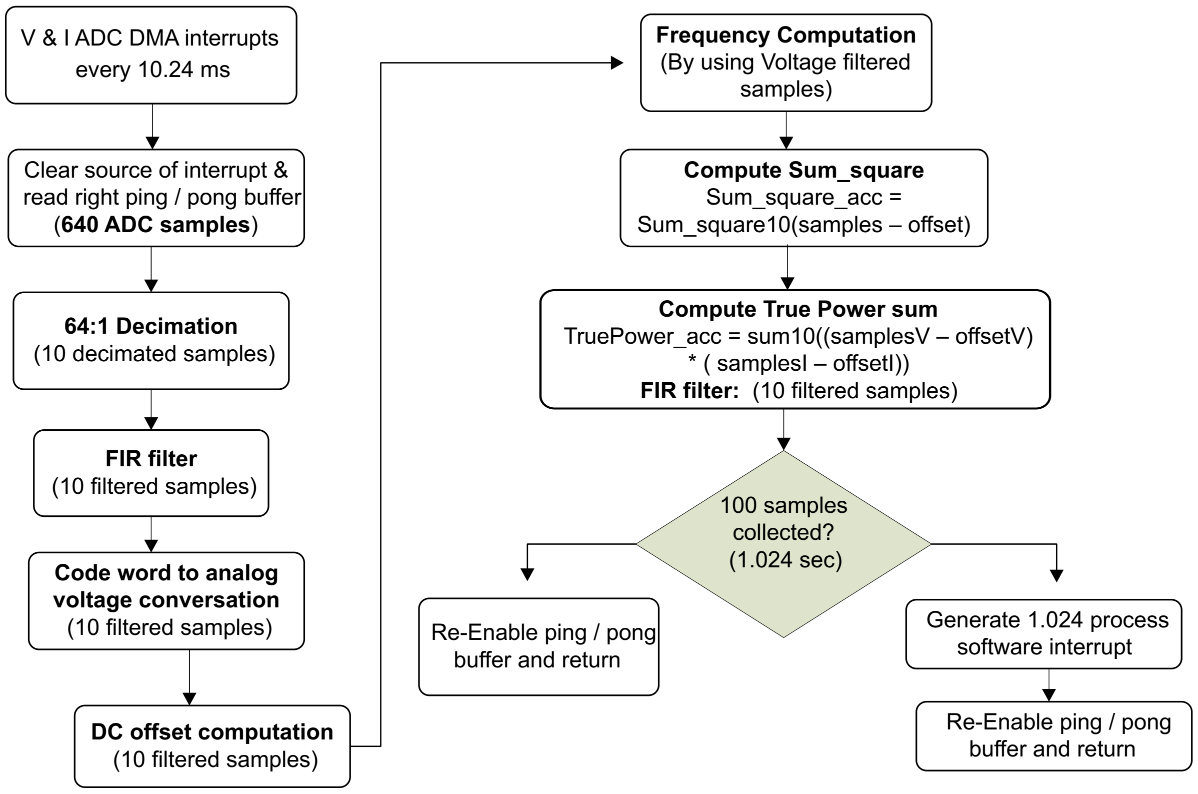

V & I ADC DMA interrupts every 10.24 ms
Clear source of interrupt & read right ping / pong buffer (640 ADC samples)
64:1 Decimation (10 decimated samples)
FIR filter (10 filtered samples)
Code word to analog voltage conversation (10 filtered samples)
DC offset computation (10 filtered samples)
Frequency Computation (By using Voltage filtered samples)
Compute Sum_square Sum_square_acc = Sum_square10(samples – offset)
Compute True Power sum TruePower_acc = sum10((samplesV – offsetV) * (samplesI – offsetI))
FIR filter: (10 filtered samples)
100 samples collected? (1.024 sec)
Re-Enable ping / pong buffer and return
Generate 1.024 process software interrupt
Re-Enable ping / pong buffer and return

Figure 19. 10.24-ms Process

3.2.6.1.1 Decimation

The 640 samples in 10.24 ms are oversampled for metrology application, so we can utilize these samples to improve the ENOB of the ADC.

The decimation improves the ENOB and quantization noise of ADC. The 64:1 decimation is used by summing 64 ADC samples (2^6) and right shifting this sum by 3 bits, which improves ENOB by 3 bits. This decimation results in a 13-bit ADC (15 code words) with 1000 samples per 1.024 sec, instead of the original 10-bit ADC (12 code words) with 64000 samples per 1.024 sec.

Decimated_adc_sample = (Σ 64 ADC samples >> 3)

Decimated_adc_sample is the 15-bit code word

3.2.6.1.2 FIR Filtering

After decimation, 10 samples pass to the 5th order FIR filter with 300Hz cutoff frequency to eliminate high-frequency harmonics and noise.

Filter coefficients of 5th order FIR filter:

b0 = -0.0105, b1 = 0.0269, b2 = 0.4836, b3 = 0.4836, b4 = 0.0269, b5 = -0.0105

Frequency response: (sampling frequency – 976.5625Hz, Cutoff Frequency – 300Hz)

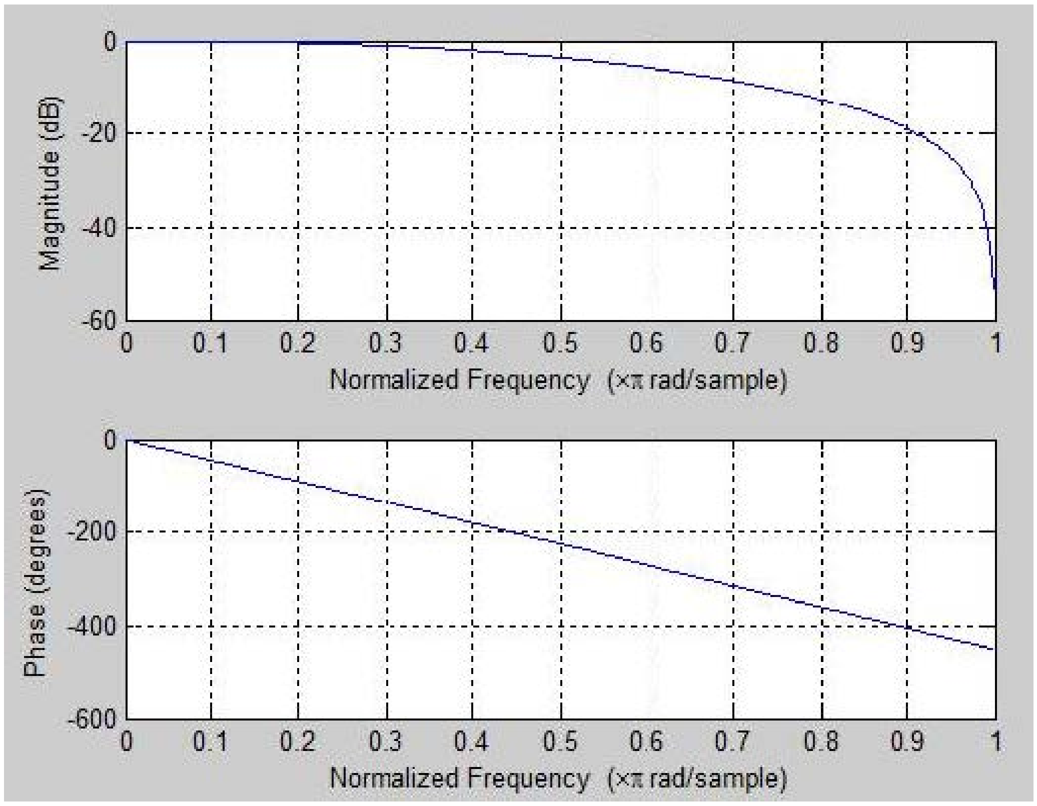


Figure 20. FIR Filtering

Filtered_adc_sample(n) = decimated_adc_sample(n) × b0 + decimated_adc_sample(n-1) × b1 + decimated_adc_sample(n-2) × b2 + decimated_adc_sample(n-3) × b3 + decimated_adc_sample(n-4) × b4 + decimated_adc_sample(n-5) × b5.

Filter coefficients are scaled by 12 bits and final filtered samples: Filtered_adc_sample(n) is a 15-bit code word. At the very first run or interrupt, all past decimated_adc_samples must be initialized to the same current value.

3.2.6.1.3 Code Word to Analog Voltage Conversion

After filtering 10 samples, convert them from a 15-bit code word to ADC dynamic range voltage format by using the following formula:

Filtered_adc_sample = Filtered_adc_sample (15-bit code word) × 1.1 × 4 / 3 × 2^{12} / 2^{15}

The output Filtered_adc_sample is in voltage format and is scaled by 12 bits.

Plots below shows the V and I channel samples at different stages: Voltage-channel-decimated and filtered samples:

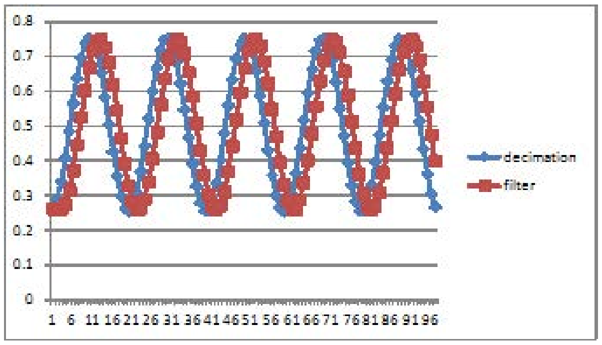


Figure 21. Voltage Channel Decimated and Filtered Samples

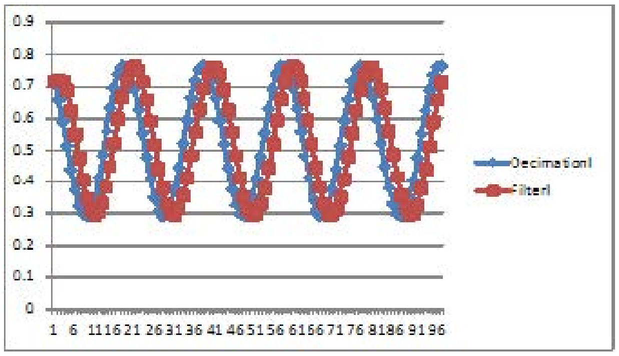


Figure 22. Current Channel Decimated and Filtered Samples

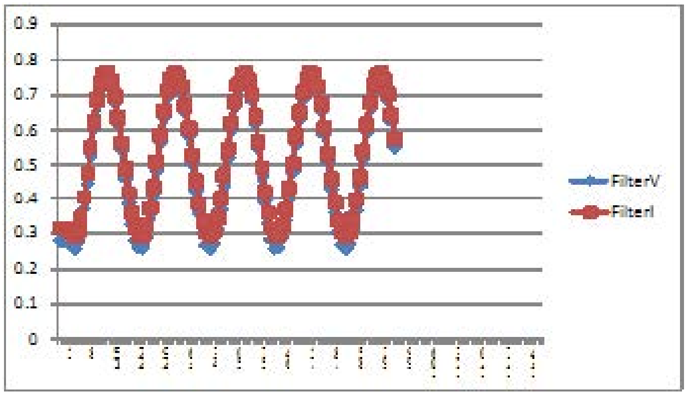


Figure 23. Filtered Voltage and Current Samples on top of Each Other with No Phase Difference After Filtering

3.2.6.1.4 Channel DC Offset Computation

Because of an initial DC offset of ~0.72 V in both V and I channels ADC samples, one must measure the DC offset accurately for further processing frequency, RMS, and power / energy computations.

The DC offset computation is done by using integral number of cycle's samples in 1.024 sec to avoid a residual affect in offset. The integral number of cycles in 1.024 sec is obtained by frequency computation logic (discussed in next section).

In 10.24-ms process, the accumulation of filtered samples happens for an integral number of cycles provided by frequency computation logic. In 10.24 ms, only 10 samples are available to integrate. In each 10.24 ms, the DC offset of incoming samples is computed and provided for further processing. In the first epoch or interrupt, the present partially averaged DC offset (which may not be accurate) is used for processing of the signal's voltage or current. In subsequent epochs or interrupts, the DC offset (which is accurate) of the last 1.024-sec epoch is used for processing.

By default, the first integral number of cycles is not known for the first 1.024 sec, so all 1000 samples (1.024-sec samples) are used to compute DC offset until the frequency computation gives the exact number of cycles per sec.

Both DC offset computation and frequency computations are dependent on each other for accurate measurement.

Channel_dc_offset = (Σ N filtered samples) / N

N = Number of samples in integral number of cycles in 1.024 sec.

To improve the accuracy of the DC offset, a 3-sample moving average DC offset value has been computed by using the past two values (1.024-sec-apart samples). Channel_dc_offset is scaled by 12 bits.

3.2.6.1.5 Supply Voltage Frequency Computation

The frequency of incoming voltage channel samples is computed by using a zero-crossing mechanism; in one cycle would be two zero crossings. All filtered samples until the detection of the first zero crossing are discarded in counting, and all filtered samples in complete half cycles in 1.024 sec are counted; the residual samples after the detection of the last zero crossing are also discarded. This count of filtered samples in total complete half cycles is required to compute fractional frequency.

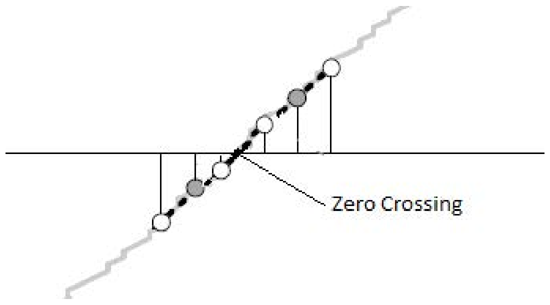


Figure 24. Computing Number of Zero Crossings and Total Samples

Zero crossing can be detected by comparing (filtered_adc_sample(n) – Channel_dc_offset) with zero. If (filtered_adc_sample(n) – Channel_dc_offset) ≥ 0, then the sample is in a positive cycle; otherwise, the sample is in a negative cycle. The transition from +ve to –ve or vice versa gives zero crossing; the number of zero crossing count in 1.024 sec gives the number of complete half cycles in 1.024 sec. The number of filtered samples in total complete half cycles is counted and helps derive the average number of samples per one full cycle. Every 10.24 ms, zero crossing detection happens for 10 samples, and the result will be accumulated for 1.024 sec.

Zero_crossing_count – total complete half cycles in 1.024 sec

Total_samples_per_integral_cycles – total filtered samples in total complete half cycles of 1.024 sec.

The previous two parameters are the result of a 10.24-ms process in 100th interrupt; the actual frequency computation will be done in a 1.024-sec process.

3.2.6.1.6 Sum Square Computation of Voltage and Current Signal

After computing the DC offset, compute the metrology parameters. RMS computation is the root mean square of the samples. To take the mean of the square samples, accumulate the square of each filtered samples. In 10.24 ms, 10 filtered samples will be accumulated; at the end of 1.024 sec (100th interrupt), the total accumulated value will be used to compute the raw voltage mean square. In 1.024 sec, 1000 samples are available for accumulation.

$$V_{RMS} = K_v \bullet \sqrt{\frac{\sum_{n=1}^{\text{Sample count}} V^2(n)}{\text{Sample count}}}$$

$$I_{RMS} = K_i \bullet \sqrt{\frac{\sum_{n=1}^{\text{Sample count}} i^2(n)}{\text{Sample count}}} \quad (6)$$

Raw_Voltage_mean_square = (Σsquare (1000 Voltage filtered samples – Voltage DC offset)) / 1000

Raw_Current_mean_square = (Σsquare (1000 Current filtered samples – Current DC offset)) / 1000

In the 1.024-sec process, RMS value will be computed from raw voltage and current mean square values. The raw value is scaled by 20 bits.

3.2.6.1.7 Dot Product Sum of Voltage and Current Signal

The true energy is the mean of the dot product of corresponding current and voltage samples; to take the mean of the dot product samples, accumulate dot product filtered samples. In 10.24 ms, 10 filtered samples will be accumulated; at the end of 1.024 sec (100th interrupt), the total accumulated value will be used to compute the raw power value. In 1.024 sec, 1000 samples are available for accumulation.

$$P_{ACT} = K_p \times \frac{\sum_{n=1}^{\text{Sample count}} v(n) \times i(n)}{\text{Sample count}} \tag{7}$$

Raw_True_Power = (Σ(1000 Voltage filtered samples – Voltage DC offset) × (1000 Current filtered samples – Currrent DC offset)) / 1000

In 1.024-sec process, the true power value will be computed from the raw power value. The raw value is scaled by 20 bits.

3.2.6.2 1.024-sec Process

The 1.024-sec process is generated at the 100th interrupt of 10.24 ms by software interrupt. This process uses the raw values generated by the 10.24-ms process to compute final metrology parameters. The validation of computed metrology parameters happens in this process.

The DC offset computed for V and I channels at the end of the 10.24-ms process is validated against the threshold value (0.5 V). If offset is less than 2048 (0.5 scaled by 12 bits), then invalidate metrology parameters and initialize all parameters to zero.

If DC offset is above the threshold, then compute all metrology parameters.

3.2.6.2.1 Metrology Parameters Computation

3.2.6.2.1.1 RMS Voltage and Current

The raw V and I mean square values are scaled to 20 bits. The RMS values can be computed by taking the root of the raw values and converting the result.
Rms_Voltage = (float) root(Raw_Voltage_mean_square << 4) / 4096
Rms_Current = (float) root(Raw_Current_mean_square << 4) / 4096
Remove RMS ADC noise from RMS values, these noise values for Voltage and Current Channels available in Nonvolatile memory stored during the first time boot calibration (calibration is discussed in the next section).
Rms_Voltage = Rms_Voltage - Rms_Voltage_channel_noise
Rms_Current = Rms_ Current - Rms_ Current _channel_noise
Multiply scaling factor of V and I channels into RMS Voltage and Current, these scaling factors are stored in nonvolatile memory during factory bring up.
Rms_Voltage = Rms_Voltage × Voltage_channel_scale_factor
Rms_Current = Rms_ Current × Current _channel_ scale_factor

3.2.6.2.1.2 True Power

The raw true power value is 20-bit scaled. The true power can be computed by down scaling the raw value.
True_Power = ((Raw_True_Power + (1 << 7))>>8)/4096
Multiply scaling factor of V and I channels power.
True_Power = True_Power × Voltage_channel_scale_factor × Current _channel_ scale_factor

Remove ADC noise power from Power value, the noise power is product of Rms_Voltage_channel_noise and Rms_ Current _channel_noise.
True_Power = True_Power – Noise_Power

3.2.6.2.1.3 Frequency

The Zero_crossing_count and Total_samples_per_integral_cycles per 1.024 sec is provided by the 100th interrupt of the 10.24-ms process. In 1.024 sec, 1000 filtered samples are present, in which only Total_samples_per_integral_cycles samples are present in a complete integral number of cycles. The difference will give residual samples in 1.024 sec, which is required to find the residual accurate frequency.

Number_of_Residual_Samples = 1000 - Total_samples_per_integral_cycles (in 1.024 sec)

The accurate number of samples per one complete cycle is computed by using Zero_crossing_count and Total_samples_per_integral_cycles, which is required to compute the residual accurate frequency.

Num_Samples_per_cycle = (Total_samples_per_integral_cycles/Zero_crossing_count) × 2

In one cycle, 2 zero crossings will be detected. Num_Samples_per_cycle is averaged over 2 samples (1.024-sec samples) for better accuracy.

The Frequency can be computed as below, which is computed by using 1.024 sec so it is necessary to convert frequency into 1-sec measuring interval.

- Frequency = (Zero_crossing_count/2 + Number_Residual_Samples/Num_Samples_per_cycle)/1.024
- Integral number of cycles per sec: Number of integral cycles per seconds is necessary to compute accurate DC offset, it can be done by using Frequency of supply voltage.
- Total_Number_of_Cycles_per_1.024sec_in_float = 1000 / Num_Samples_per_cycle
- Total_integral_cycles_per_1.024sec = (int) Total_Number_of_Cycles_per_1.024sec_in_float
- Residual_samples = (int) ((Total_Number_of_Cycles_per_1.024sec_in_float - Total_integral_cycles_per_1.024sec) × Num_Samples_per_cycle + 0.5)

The Total samples in integral cycles in 1.024 sec used for DC computation is as below

- Total_samples_in_integral_cycles_per_1.024sec = 1000 - Residual_samples

3.2.6.2.1.4 Apparent Power

Apparent power is a product of RMS voltage and current.

$$P_{APT} = V_{RMS} \bullet I_{RMS} \tag{8}$$

Apparent_Power = Rms_Voltage × Rms_Current

3.2.6.2.1.5 Reactive Power

Reactive power can be computed by using true active power and apparent power.

$$P_{APT}{}^{2} = P_{ACT}{}^{2} + P_{REACT}2 \tag{9}$$

Reactive_Power = sqrt(Apparent_Power × Apparent_Power - True_Power × True_Power)

True Active Energy

True active energy is the accumulation of true power in every 1 sec. Because we compute power every 1.024 sec, the energy spent in 1.024 sec is 1.024 × True_Power.

True_Energy = Σ(True_Power × 1.024)

True_energy_kWh = True_Energy / 1000 / 3600 kWh

3.2.6.2.2 ADC Channel Noise Calibration

The noise in the current ADC channel can be computed by switching off the load and measuring the Rms_Current, which directly gives RMS noise in the current ADC channel. We are assuming by design that both current and voltage ADC must have the same characteristics and that the noise in both channels is identical.

3.2.6.2.3 ADC Channels Scaling Factor

The scaling factor of voltage and current channels can be computed during the factory-bring-up calibration by connecting known load and supply to device, The Android app has a provision to perform the calibration of the CC3200_Smart_Plug.

3.3 Embedded Network Software Module

The CC3200_Smart_Plug network software is developed based on FreeRTOS. FreeRTOS is based on tasks that are added to the schedule by the application to run at specific priorities to facilitate TCP communication. These tasks leverage the network stacks provided by the CC3200 development environment to simplify the communication. The following sections will outline the critical tasks associated with the CC3200_Smart_Plug application code. The high-level flow diagram of CC3200_Smart_Plug network software module is shown in Figure 25.

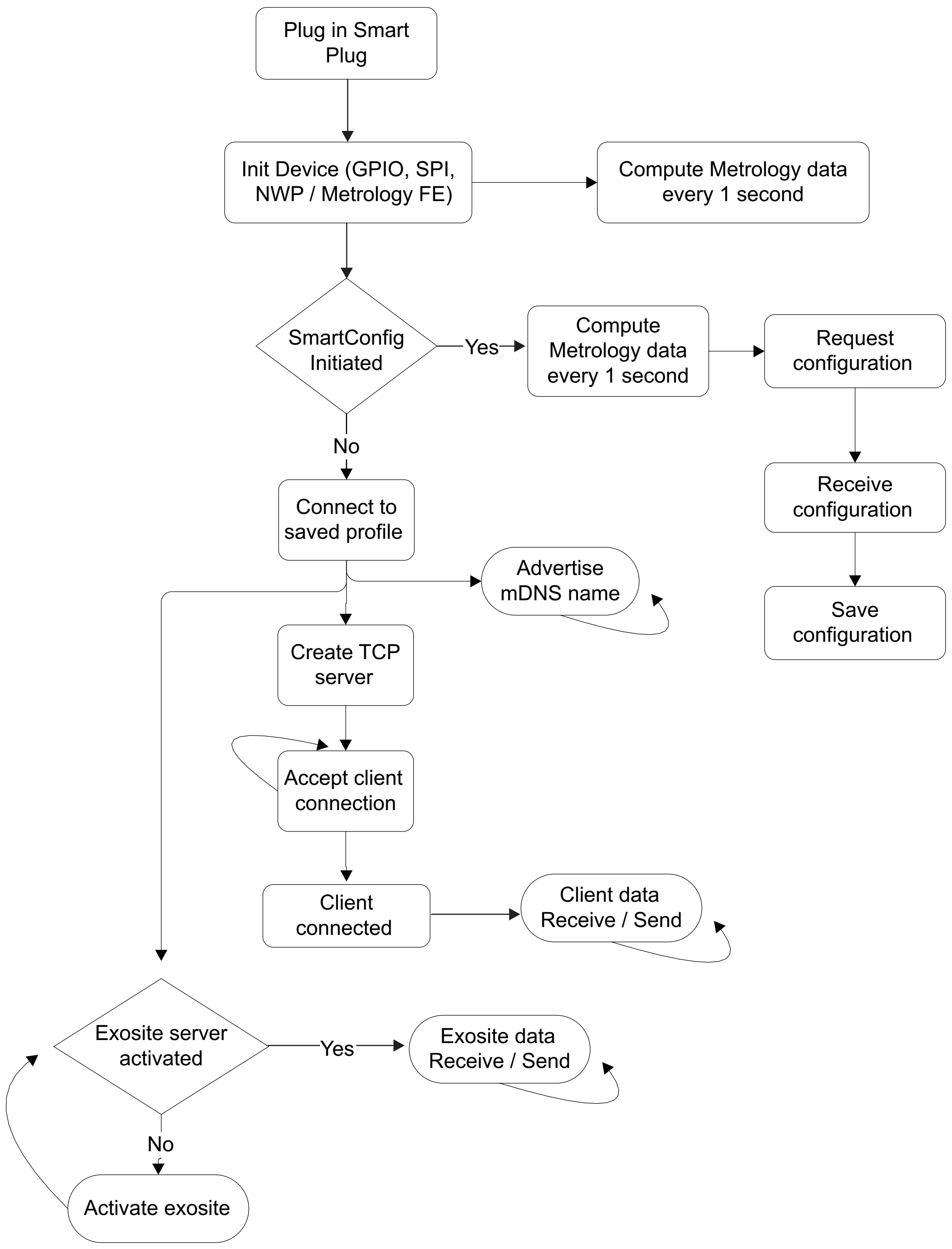


Figure 25. Embedded Network Software Module

3.3.1 Main Function

The CC3200_Smart_Plug main function sets up the CC3200 system hardware and CPU. This function is broken into a simplified flow chart as shown in Figure 26.

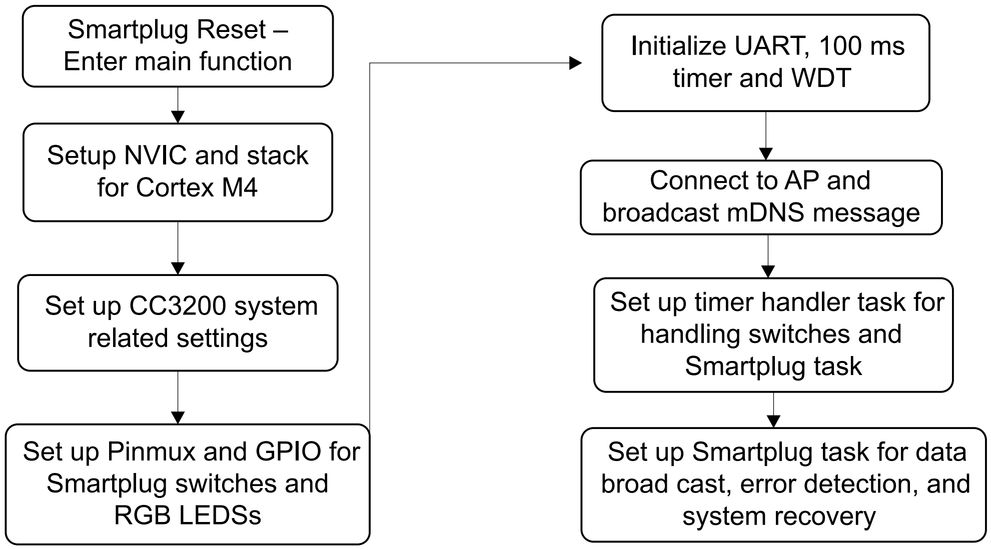

Smartplug Reset – Enter main function
Setup NVIC and stack for Cortex M4
Set up CC3200 system related settings
Set up Pinmux and GPIO for Smartplug switches and RGB LEDSs
Initialize UART, 100 ms timer and WDT
Connect to AP and broadcast mDNS message
Set up timer handler task for handling switches and Smartplug task
Set up Smartplug task for data broad cast, error detection, and system recovery

Figure 26. Flow of the CC3200_Smart_Plug Main Function

3.3.2 Timer Handlers

3.3.2.1 100-ms Timer Handler Task

This task is a heartbeat interrupt for CC3200 every 100 ms to monitor the GPIO smartconfig, relay state switches, and toggle GPIO RGB status LEDs. Every 1 sec, this interrupt routine triggers the timer handler task, which is the first low-priority task of the system.

3.3.2.2 Timer Handler Task

This task does the following:

- configures the Relay state based on the GPIO input when every time switch is being pressed
- manages the CC3200_Smart_Plug time
- maintains the relay state based on the schedule table
- updates the NVMM file if in modification in the NVMM state
- triggers the periodic 1-sec CC3200_Smart_Plug task

.

3.3.3 CC3200_Smart_Plug Task

The CC3200_Smart_Plug task periodically broadcasts metrology data and handles error recovery if the network processor of CC3200 is not responding. During a first-time reset, this task initializes the entire CC3200_Smart_Plug, which has been simplified as shown in Figure 27.

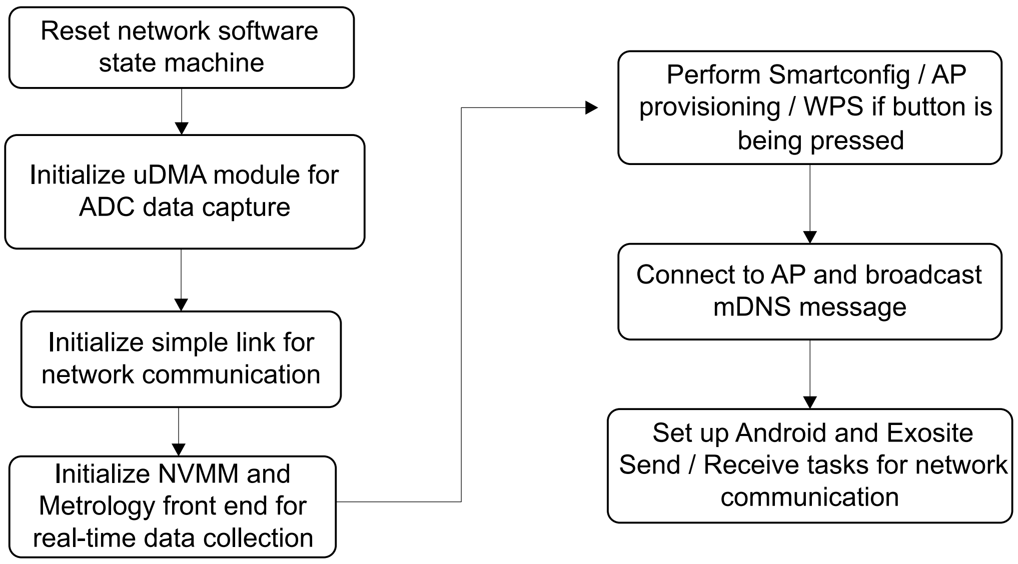

Reset network software state machine
Initialize uDMA module for ADC data capture
Initialize simple link for network communication
Initialize NVMM and Metrology front end for real-time data collection
Perform Smartconfig / AP provisioning / WPS if button is being pressed
Connect to AP and broadcast mDNS message
Set up Android and Exosite Send / Receive tasks for network communication

Figure 27. Flow of the CC3200_Smart_Plug Task Function

The periodic 1-sec CC3200_Smart_Plug task triggers the Exosite and Android send task to broadcast metrology parameters into user. The below flow chart explains this periodic functionality.

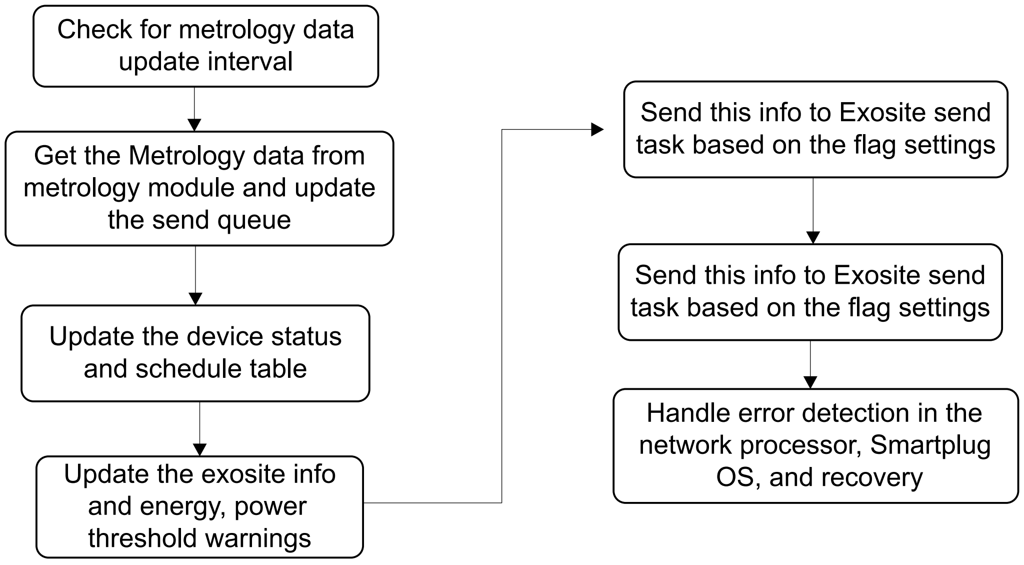

Check for metrology data update interval
Get the Metrology data from metrology module and update the send queue
Update the device status and schedule table
Update the exosite info and energy, power threshold warnings
Send this info to Exosite send task based on the flag settings
Send this info to Exosite send task based on the flag settings
Handle error detection in the network processor, Smartplug OS, and recovery

Figure 28. Flow of Periodic Triggers to Send the Data to the Cloud (Exosite) and Mobile App

3.3.4 Exosite Tasks

3.3.4.1 Send Task

The Exosite Send task transfers metrology data from the metrology module to the Exosite server by using TCP connection. The CC3200_Smart_Plug task triggers this task at every report interval. Figure 29 explains the flow of this task.

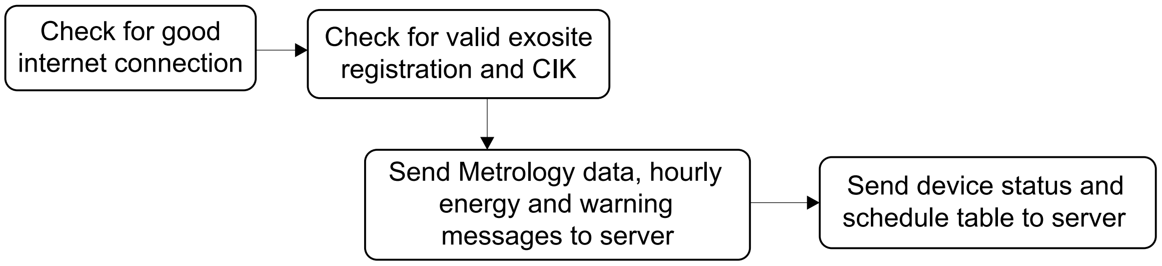

Check for good internet connection
Check for valid exosite registration and CIK
Send Metrology data, hourly energy and warning messages to server
Send device status and schedule table to server

Figure 29. Exosite Send Task

3.3.4.2 *Exosite Receive Task*

The Exosite Receive task reads the device status from the server by using TCP connection. The CC3200_Smart_Plug network processor triggers this task whenever the processor is free. Figure 30 explains the flow of this task.

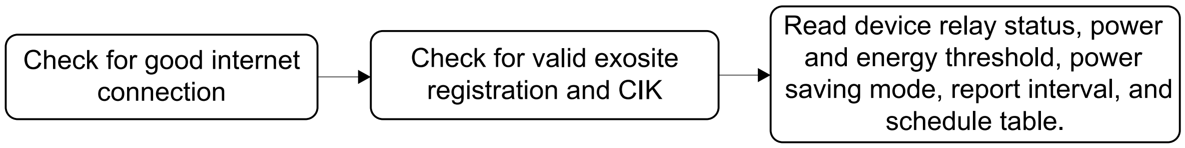

Check for good internet connection
Check for valid exosite registration and CIK
Read device relay status, power and energy threshold, power saving mode, report interval, and schedule table.

Figure 30. Exosite Receive Task

3.3.5 Android Tasks

3.3.5.1 *Android Send Task*

The Android Send task transfers metrology data from the metrology module to the Android client by using TCP connection. The CC3200_Smart_Plug task triggers this task at every report interval. Figure 31 explains the flow of this task.

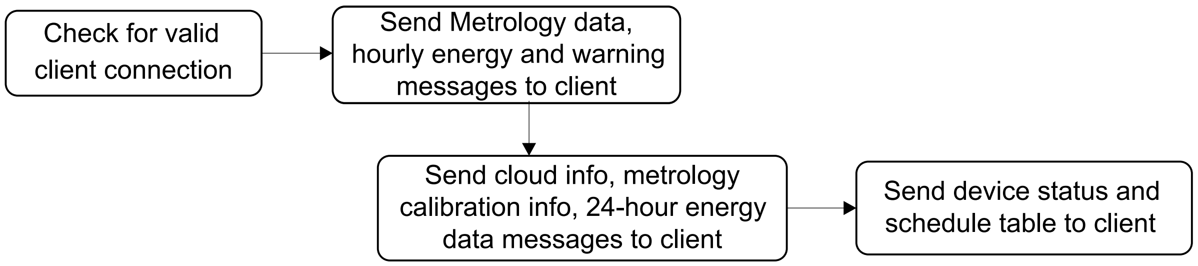

Check for valid client connection
Send Metrology data, hourly energy and warning messages to client
Send cloud info, metrology calibration info, 24-hour energy data messages to client
Send device status and schedule table to client

Figure 31. Android Send Task

3.3.5.2 *Android Receive Task*

The Android Receive task reads the device status from the client by using TCP connection. The CC3200_Smart_Plug network processor triggers this task whenever the processor is free and a client is connected to CC3200_Smart Plug. Figure 32 explains the flow of this task.

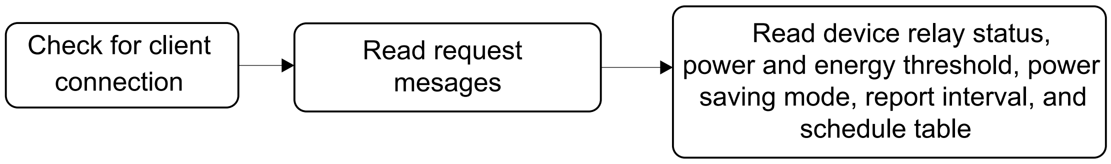

Check for client connection
Read request mesages
Read device relay status, power and energy threshold, power saving mode, report interval, and schedule table

Figure 32. Android Receive Task

3.4 Packet Structure

The following packet structures communicate with the Android client and Exosite server.

3.4.1 Device to Android

Table 2. Packet to Update Metrology Data

BYTE	0 – 1	2 – 5	6 – 9	10 – 13	14 – 17	18 – 21	22 – 25	26 – 29	30 – 33	34 – 37	38 – 41
DATA	0 × 7A01	Power (W)	Voltage (V)	Current (A)	Frequency (Hz)	VAR	Cos()	VA	KWh	Avg Power (W)	Time stamp
FORMAT	U16	U32	U32	U32	U32	U32	U32	U32	U32	U32	U32
RESOLUTION	1	0.001 (data × 1000)	0.001	0.001	0.001	0.001	0.0001 (data × 10000)	0.001	0.0001	0.001	1

Android communication byte order:

All the send and receive data are in little endian format except the header. (Example: see Table 2 packet)

Table 3. Example of an Android Communication Byte Order

BYTE	0 – 1	2 – 5
DATA	0 × 7A01	Power (W) 5000 = 0 × 00001388
PACKET ORDER	0 × 7A, 0 × 01	0 × 88, 0 × 13, 0 × 00, 0 × 00

The packet Android app would receive 0 × 7A, 0 × 01, 0 × 88, 0 × 13, 0 × 00, and so on. The same rule would apply for packets received from Android.

Table 4. Packet to Update Average Energy at Hourly Frequency

BYTE	0 – 1	2 – 5	6 – 9
DATA	0 × 7A02	Average Energy (kWh)	Timestamp
FORMAT	U16	U32	U32
RESOLUTION	1	0.0001 (data × 10000)	1

Table 5. Packet to Update Device Status

BYTE	0 – 1	2	3	4	5 – 8	9 – 12	13 – 16
DATA	0 × 7A04	Device turned Off / On (0/1)	Power saving mode Enable/Disable (1 / 0)	Update Interval in sec 5 < X < 45 secs, incr = 5	Energy threshold value (KWh)	Power threshold Value (W)	Timestamp
FORMAT	U16	U8	U8	U8	U32	U32	U32
RESOLUTION	1	1	1	1	0.0001 (data × 10000)	0.001 (data × 1000)	1

Table 6. Warning Message for Over Energy and Power Consumption Threshold

BYTE	0 – 1	2	3	4	5 – 8
DATA	0 × 7A08	Over Energy (kWh) consumption	Over Power (W) consumption	Metrology module error	Timestamp
FORMAT	U16	U8	U8	U8	U32
RESOLUTION	1	1	1	1	1

NOTE: User must clear the warning message in the app.

Table 7. Average Energy (KWh) of Last 24 Hours

BYTE	0 – 1	2 – 5	6 – 9	10 – 13	……	94 – 97	98 – 101
DATA	0 × 7A10	Average Energy stored of last hour	Average power of past 1-2 hour	Average power of past 2-3 hour	…….	Average power of past 23-24 hour	Timestamp
FORMAT	U16	U32	U32	U32		U32	U32
RESOLUTION	1	0.0001 (data × 10000)	0.0001	0.0001		0.0001	1

Table 8. Device's Default Switch Table

BYTE	0 – 1	2 – 25				26 – 28	29
DATA	0 × 7A20	Switch table Weekdays (M – F) and Weekend (Sa and Su) – 7days Various times: Wake up, leave, return, sleep for 3 days (week days and sat, sun) settings				CC3200 Smart Plug Time	Schedule table validity
		Wakeup (ON)	Leave (OFF)	Return (ON)	Sleep (OFF)		Active/Not active (1 / 0)
		Hour:Min	Hour:Min	Hour:Min	Hour:Min	Day:Hour:Min	
FORMAT	U16	U8:U8	U8:U8	U8:U8	U8:U8	U8:U8:U8	U8
RESOLUTION	1	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Day: 0 (Sun) – 6 (Sat) Hour: 0 – 23 Min: 0 – 59	1

NOTE: Table validity bit indicates whether the table is currently active or not.

Table 9. Cloud Specific Requirements

BYTE	0 – 1	2 – 18	19 – 30	31 – 36	37	38 – 41
DATA	0 × 7A40	Vendor's name: texasinstruments	Model: smartplugv2	Mac Address	Bit 0: Indication of Exosite initialized in CC3200 Smart Plug. Bit 1: Connected to cloud (TCP communication is active) Bit 2: Secured connection Bit 3: Indication of valid vendor information/MAC address in Exosite server. Bit 4: Indication of valid usable CIK in CC3200 Smart Plug (active registration)	Timestamp
FORMAT	U16	String-17char	String-12char	U8:U8:U8:U8:U8:U8	U8	U32
RESOLUTION	1	string	string	1	1	1

Table 10. Metrology Calibration Specific Requirements

BYTE	0 – 1	2 – 5	6 – 9	10 – 13	14 – 17	18 – 21
DATA	0 × 7A80	Voltage Channel noise	Current Channel noise	Voltage Channel Scale factor	Current Channel Scale factor	Timestamp
FORMAT	U16	U32	U32	U32	U32	U32
RESOLUTION	1	0.00001 (data × 100000)	0.00001 (data × 100000)	0.00001 (data × 100000)	0.00001 (data × 100000)	1

Table 11. Message to Indicate Success in Updating Exosite SSL CA Certificate

BYTE	0 – 1	2 – 5
DATA	0 × 7A91	Timestamp
FORMAT	U16	U32
RESOLUTION	1	1

If write is successful, then the app can enable a secured Exosite connection (**Table 22**) to validate the certificate.

3.4.2 Android to Device

Table 12. Packet to Turn Off / On Connected Device (Relay)

BYTE	0 – 1	2
DATA	0 × A701	Turn Off/On (0/1)device
FORMAT	U16	U8
RESOLUTION	1	1

NOTE: The user modification of the current state of relay will supersede the value of the schedule table configuration in the CC3200 Smart Plug. The schedule-table-validity bit will be cleared upon a change in state of Relay by this message.

Table 13. Packet to Set Energy and Power Threshold for Warning

BYTE	0 – 1	2 – 5	6 – 9
USE	0 × A702	Energy threshold value (KWh)	Power threshold Value (W)
FORMAT	U16	U32	U32
RESOLUTION	1	0.0001 (data × 10000)	0.001(data × 1000)

Table 14. Packet to Set Power-Saving Mode

BYTE	0 – 1	2	3
USE	0 × A704	Disable/Enable Power saving mode (0/1)	Time interval for periodic update in seconds
FORMAT	U16	U8	U8
RESOLUTION	1	1	1

Table 15. Packet to Request Past 24 Hours Average Power

BYTE	0 – 1
USE	0 × A708

Table 16. Packet to Request Device's Default Switch Table

BYTE	0 – 1
USE	0 × A710

Table 17. Packet to Request Metrology Calibration Information

BYTE	0 – 1
USE	0 × A720

Table 18. Packet to Request Cloud Specific Requirement

BYTE	0 – 1
USE	0 × A740

Table 19. Packet to Request Device Information (Device On / Off and Power-Save Mode)

BYTE	0 – 1
USE	0 × A780

Table 20. Update Device's Switch Table

BYTE	0 – 1	2 – 25				26 – 28	29
DATA	0 × A791	Switch table Weekdays (M – F) and Weekend (Sa and Su) – 7days Various times: Wake up, leave, return, sleep for 3 days (week days and sat, sun) settings				Synch Time	Activate Schedule table
		Wakeup(ON)	Leave(OFF)	Return(ON)	Sleep(OFF)		Enable/Disable (1 / 0)
		Hour:Min:	Hour:Min	Hour:Min	Hour:Min	Day:Hour:Min	
FORMAT	U16	U8:U8	U8:U8	U8:U8	U8:U8	U8:U8:U8	U8
RESOLUTION	1	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Day: 0 (Sun) - 6 (Sat) Hour: 0 – 23 Min: 0 – 59	1

NOTE: Send non-zero values for effective use of schedule table in CC3200 Smart Plug, if all values are zero (wakeup, leave, return, sleep = 0) then CC3200 Smart Plug will not use schedule table.

Table 21. Update Metrology Calibration Specific Information

BYTE	0 – 1	2 – 5	6 – 9
DATA	0 × A792	Desired Voltage	Desired Current
FORMAT	U16	U32	U32
RESOLUTION	1	0.00001 (data × 100000)	0.00001 (data × 100000)

NOTE: See Section 3.6 for the calibration procedure.

Table 22. Packet to Enable or Disable Secured Exosite Connection

BYTE	0 – 1	2
DATA	0 × A794	Enable / Disable secured Exosite connection (1 / 0)
FORMAT	U16	U8
RESOLUTION	1	1

Table 23. Packet to Update Exosite SSL Client CA Certificate

BYTE	0 – 1	2 – 3	4 – N	N+1
DATA	0 × A798	Certificate length	Certificate Binary data	checksum
FORMAT	U16	U16	U8	U8
RESOLUTION	1	1	1	1

The checksum is just XOR of all the bytes of certificate (4-N bytes) – during this large data transfer (more than 1KB), the Android app can change the update rate of the CC3200 Smart Plug to the maximum value to give full bandwidth to this operation. The CC3200 Smart Plug closes the Exosite socket while updating this certificate. The certificate must be in binary format (.der) before it can be sent to the Smart Plug.

3.4.3 Device to Server

Table 24. Packet to Update Metrology Data

DATA	POWER	VOLTAGE	CURRENT	FREQUENCY	VAR	COS()	VA	KWh	Avg Power (W)
ALIAS	“power”	“volt”	“current”	“freq”	“var”	“pf”	“va”	“kwh”	“pow_avg”
FORMAT	Float	float	float	float	float	float	float	float	float

Plot: pow_avg and power

Display: all data in dash board

Table 25. Packet to Update Average Energy at Hourly Frequency (Update Rate 1 Hour)

DATA	AVERAGE ENERGY (KWH)
ALIAS	“enr_avg”
FORMAT	float

Display: data in dash board

Table 26. Packet to Update Device Information (Relay and PS Mode Info)

DATA	DEVICE TURNED OFF / ON (0 / 1)	POWER SAVING MODE ENABLE DISABLE (0 / 1)	UPDATE INTERVAL IN SEC	ENERGY THRESHOLD VALUE (KWH)	POWER THRESHOLD VALUE (W)
ALIAS	“control”	“powmode”	“reportint”	"enthld"	“powthld”
FORMAT	integer	Integer	Integer	float	float

Display: all data in dash board

Table 27. Warning Message for Over Energy Consumption

DATA	Over Energy(kWh) consumption
ALIAS	“enr_mesg = energy_threshold_exceeded”
FORMAT	string

Display: data in dash board, user must clear warning messages in cloud

Table 28. Warning Message for Over Power Consumption

DATA	Over Power (W) consumption
ALIAS	“pow_mesg = power_threshold_exceeded”
FORMAT	string

Display: data in dash board, user must clear warning messages in cloud

Table 29. Device’s Default Schedule Table

DATA	Schedule Weekdays (M – F) and Weekend (Sat and Sun) – 7days Various times: Wake up, leave, return, sleep for 3 days (week days and week end) settings				CC3200 Smart Plug Time (local time)	Schedule table validity
	Wakeup (ON)	Leave (OFF)	Return (ON)	Sleep (OFF)		Active/Not-active (1 / 0) indication
	Hour:Min	Hour:Min	Hour:Min	Hour:Min	Day:Hour:Min	
CSV STRING	102	102	102	102	60102	1
FORMAT	string	string	string	string	string	string
RESOLUTION	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Day: 0 (Sun) – 6 (Sat) Hour: 0 – 23 Min: 0 – 59	Validity- 00/01

Display: all data in dash board and time zone of CC3200 Smart Plug
In HTTP content body the string looks like the following:
Schedule = 0630, 0815, 1900, 2300,….,2300, 002130, 01 (configuration for weekdays + Saturday + Sunday (3) days)

NOTE: The table validity bit indicates whether the table is currently active or not.

Table 30. Packet to Indicate Data Port Modified

DATA	Indication of schedule or other settings update
ALIAS	"command"
FORMAT	integer

Exosite will set “command = 1” when any of the following have changed by the user via the cloud:

- schedule
- powmode
- reportint
- powthld
- repthld

CC3200 Smart Plug sends data "command = 0" in order to notify Exosite that it received the updated schedule and other settings.

3.4.4 Server to Device

The device will read all data packets, except schedule, together in one read. The settings-update indication (command) should be part of this read.

Table 31. Packet to Turn Off / On Connected Device (Relay)

DATA	DEVICE TURNED OFF / ON (0/1)
ALIAS	"control"
FORMAT	integer

NOTE: If device status “control” modifies the current state of relay, then it will override the schedule table configuration in the CC3200 Smart Plug. The schedule table validity bit will be cleared upon a change in state of Relay by this message.

Table 32. Packet to Set Energy and Power Threshold

DATA	ENERGY THRESHOLD VALUE (KWh)	POWER THRESHOLD VALUE (W)
ALIAS	"enthld"	“powthld”
FORMAT	float	float

Table 33. Packet to Set Power-Saving Mode

DATA	DISABLE/ENABLE POWER SAVING MODE (0/1)	TIME INTERVAL FOR PERIODIC UPDATE IN SECONDS
ALIAS	"powmode"	“reportint”
FORMAT	integer	integer

Table 34. Packet to Indicate Update in Schedule

DATA	Indication of schedule or other settings update
ALIAS	"command"
FORMAT	integer

If "command = 1", read all messages from Exosite server (indicates Exosite updated schedule or other settings for the CC3200 Smart Plug). Once the schedule is received, the CC3200 Smart Plug sends "command = 0" (Table 30)

Table 35. Update Device's Schedule

DATA	Schedule Weekdays (M – F) and Weekend (Sa and Su) – 7days Various times: Wake up, leave, return, sleep for 3 days (week days and week end) settings				Synch Time (Local time of CC3200 Smart Plug)	Activate Schedule table
	Wakeup(ON)	Leave (OFF)	Return (ON)	Sleep (OFF)		Enable/Disable (1 / 0) Table
	Hour:Min	Hour:Min	Hour:Min	Hour:Min	Day:Hour:Min	
CSV STRING	102	102	102	102	60102	1
FORMAT	string	string	string	string	string	string
RESOLUTION	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Hour: 0 – 23 Min: 0 – 59	Day: 0(sun)-6(sat) Hour: 0 – 23 Min: 0 – 59	Validity – 00/01

This message is separate and stored in the "schedule" data source and is to be read if "command = 1". If read based on "command = 1", CC3200 Smart Plug should set "command = 0".

NOTE: Note: Send non-zero values for effective use of schedule table in CC3200 Smart Plug. If all values are zero (wakeup, leave, return, sleep = 0), then the CC3200 Smart Plug will not use the schedule table.

Example packet: basic CSV string
In an HTTP content body, the string looks like the following:
Schedule = 0630, 0815, 1900, 2300, …., 2300, 002130, 01 (configuration for weekdays + Saturday + Sunday (3 days))
This entire string should be stored in the "schedule" data source.

3.5 Smart Plug Metrology Calibration

The metrology hardware voltage and current scaling factor need to be calibrated for optimum load. A provision to store V-and-I channel scaling factors in s-flash can be updated over the air by using the Android app.

In the Android app, the packet from Table 21 must be used to update scaling factors. The Android app must also have a calibration-feature section to calibrate the smart plug in factory bring-up. The calibration, if performed again, must be done with a purely resistive load. The calibration procedure is described below:

1. Take known voltage and current-and-power input from the user (this is a one-time configuration in the lab). The known power load must be selected in the middle of the dynamic range of the smart plug (0 to 2 KW), so around ~1 KW known load must be used
2. Read the scaling factor's default values in the smart plug by using the packet from Table 10
3. Collect voltage-and-current outputs from the smart plug for 5 sec and average the data
4. Compute new V-and-I scaling factor in android app and send that to the smart plug
5. Cross verify the new voltage and current with known values

3.6 TCP Socket Timeout Requirements

3.6.1 Smart Plug Server (Server Android Client)

1. Server is waiting for client to connect: the accept api is triggered every 500 ms (non-blocking—for error recovery) to check if any client connected
2. Receive time out: 5 seconds—but the CC3200 Smart Plug does not close the socket if in timeout (timeout is just for error recovery); it will keep retrying
3. Send timeout: send is non-blocking api in SimpleLink, so no timeout; if sending error occurs, server tries 3 times but socket will not be closed.
4. Client socket closes:
   (a) If client is disconnected and receives api error = 0
   (b) If receive api returns any SSL error
   (c) When it receives an api error value other than SL_EAGAIN and POOL_IS_EMPTY
   (d) During the error recovery
5. Server socket closes:
   (a) If any error in accept api other than SL_EAGAIN, POOL_IS_EMPTY, SL_INEXE, and SL_ENSOCK
   (b) During the error recovery

3.6.2 Android Client

1. Client must try connecting to server indefinitely when the connection-status switch is "ON" in app
2. Connect time out: 5 secs, retry connection by closing and reopening socket if any error is reported
3. Receive time out: 2 seconds (twice min update rate)—but app should not close the socket if in timeout (can be used for error recovery); it should retry again
4. Send timeout: 2 seconds—In case of error in sending, the client should try 3 times, if failed after 3 attempts, close the socket and reopen
5. Socket closes:
   (a) If server closes the socket and receive api returns error = 0
   (b) If receive api returns any SSL error
   (c) When it receives and sends an error value other than EAGAIN and POOL_IS_EMPTY (if using timeout)

3.6.3 Exosite Client

1. Client must try connecting to server indefinitely if connection does not succeed—this issue could be an indication of no internet or SSL error
2. Connect timeout: 5 secs—retry connection by closing and reopening socket if any error reported
3. Receive timeout: 5 second—Receive api is called only after http POST for HTTP response receive; if there is a timeout, then try 3 times to close the socket.
   (a) Send timeout: send is non-blocking api in simplelink, so no timeout; in case of error in sending, it tries 3 times but socket will not be closed.
4. Socket closes:
   (a) If server closes and the socket receive api returns error = 0
   (b) If receive api returns any SSL error
   (c) When it receives an api error value other than EAGAIN and POOL_IS_EMPTY and also after 3 retries
   (d) When it sends an api error value other than EAGAIN and also after 3 retries

4 Flows of the CC3200 Smart Plug Design

Here are two flows of the CC3200 Smart Plug design, which address the overall communication between the CC3200 Smart Plug device, the Android app, and the cloud server. Flow diagram for the CC3200 Smart Plug embedded code

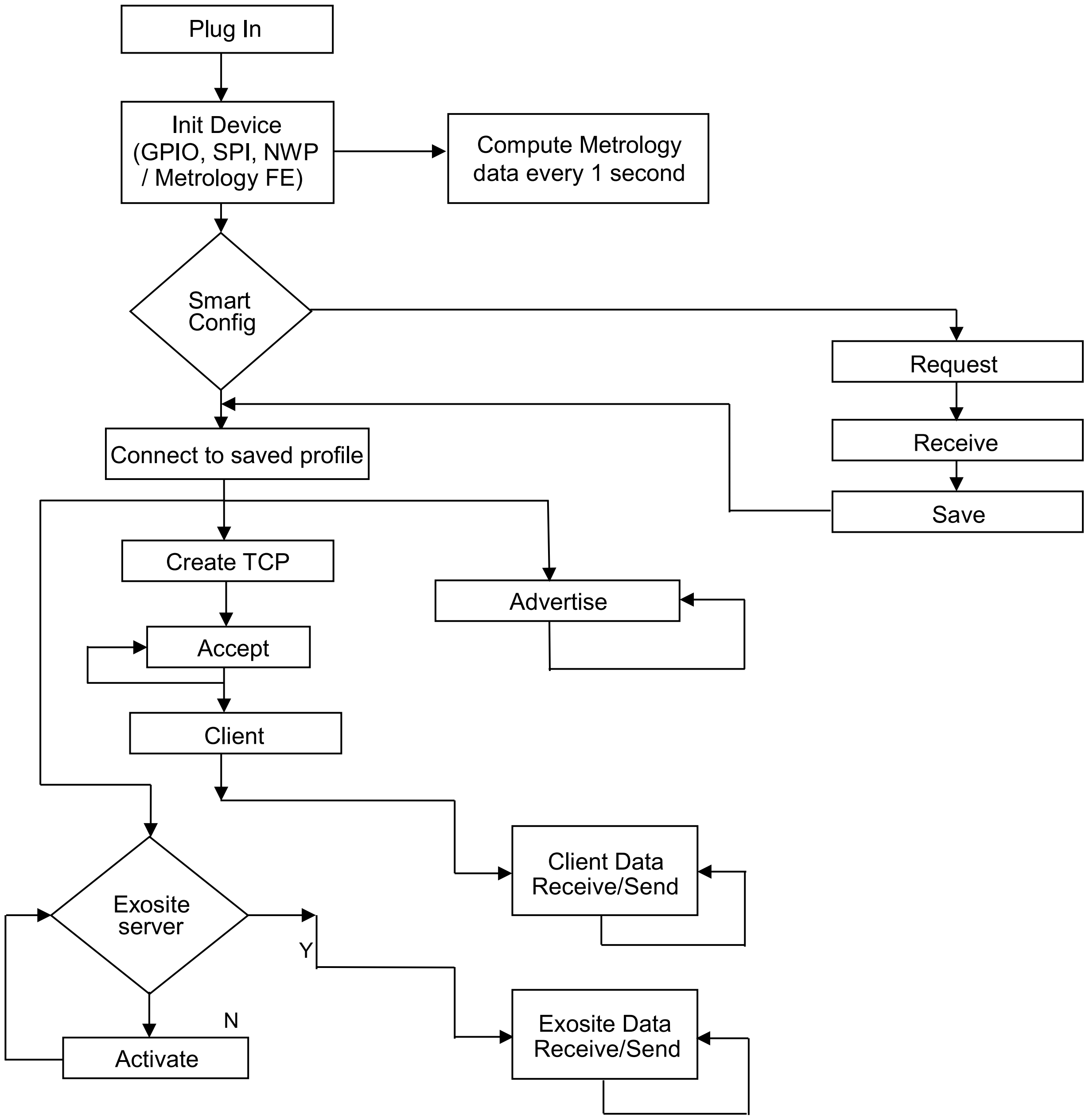


Figure 33. Flow Diagram for the CC3200 Smart Plug Embedded Code

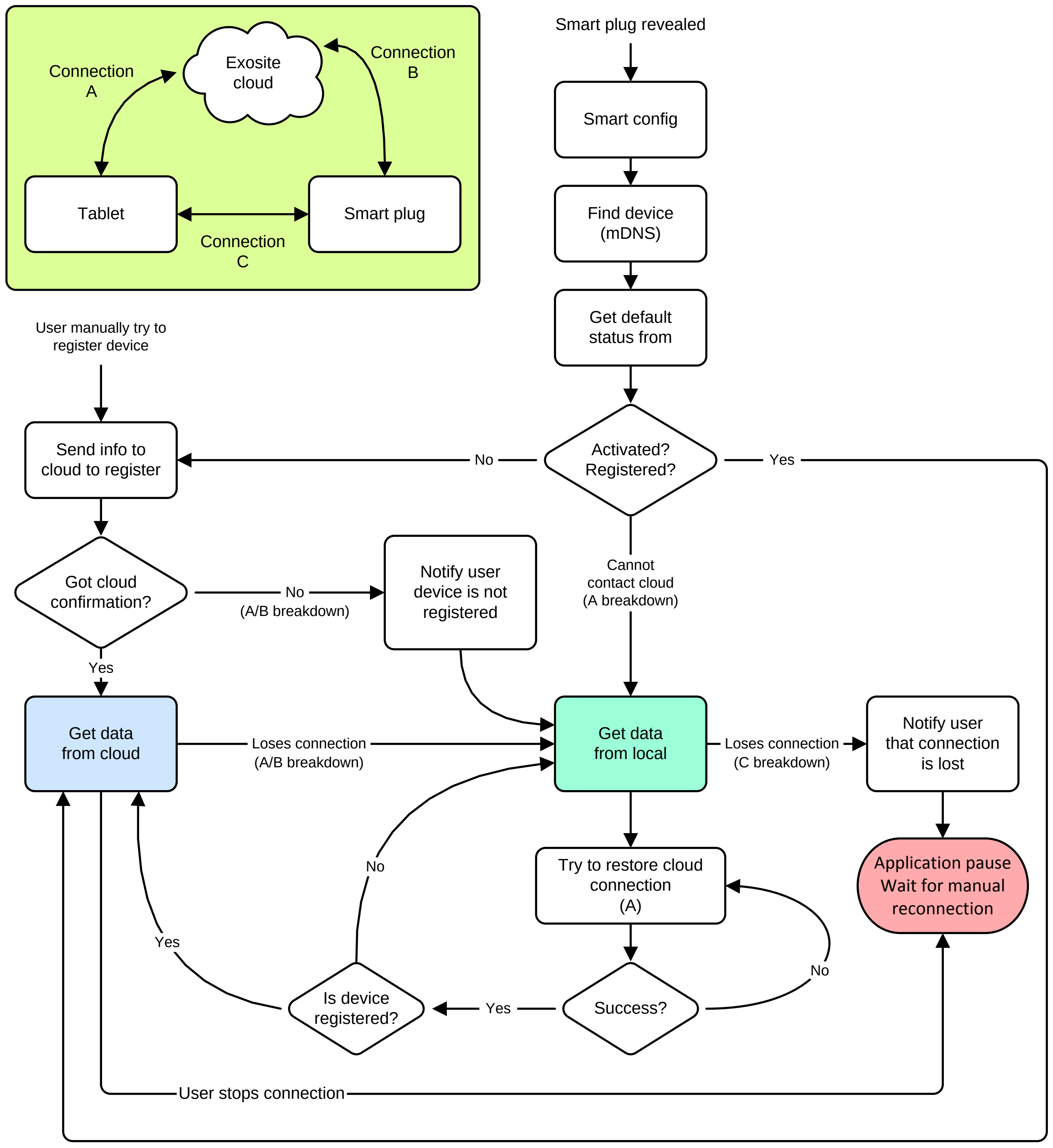
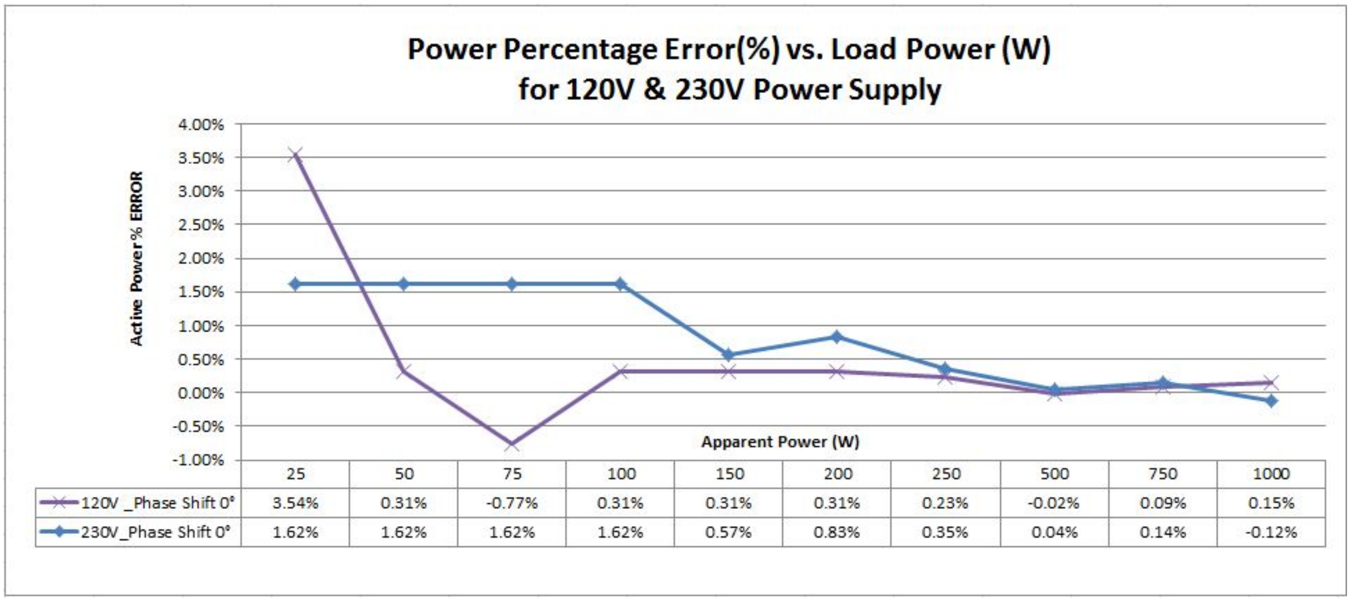

Exosite cloud
Connection A
Connection B
Tablet
Smart plug
Connection C
Smart plug revealed
Smart config
Find device (mDNS)
Get default status from
User manually try to register device
Send info to cloud to register
No
Activated? Registered?
Yes
Got cloud confirmation?
No (A/B breakdown)
Notify user device is not registered
Cannot contact cloud (A breakdown)
Yes
Get data from cloud
Loses connection (A/B breakdown)
Get data from local
Loses connection (C breakdown)
Notify user that connection is lost
Application pause Wait for manual reconnection
No
Try to restore cloud connection (A)
Yes
No
Is device registered?
Yes
Success?
User stops connection

Figure 34. Flow of the CC3200 Smart Plug from the Android App Perspective

For more information, refer to TIDU982.

5 Test Results

As shown in the application demo, the functionality of the CC3200 Smart Plug has been tested successfully. Figure 35 shows a graph of the device's energy-reading accuracy. Energy-reading accuracy shows errors below 5% across a wide dynamic range, making the CC3200 Smart Plug demo kit suitable for many applications.



	25	50	75	100	150	200	250	500	750	1000
120V _Phase Shift 0°	3.54%	0.31%	-0.77%	0.31%	0.31%	0.31%	0.23%	-0.02%	0.09%	0.15%
230V_Phase Shift 0°	1.62%	1.62%	1.62%	1.62%	0.57%	0.83%	0.35%	0.04%	0.14%	-0.12%

Figure 35. Energy-Reading Accuracy

6 Design Files

6.1 Schematics

To download the schematics for each board, please see the design files at http://www.ti.com/tool/TIDC-CC3200SMARTPLUG.

6.2 Bill of Materials

To download the bill of materials (BOM), see the design files at http://www.ti.com/tool/TIDC-CC3200SMARTPLUG.

6.3 Software Files

To download the software files, see the design files at http://www.ti.com/tool/TIDC-CC3200SMARTPLUG.

7 About the Author

This project was made possible with the collaboration of several SimpleLink™ Wi-Fi® systems and application engineers, namely **BEATRICE FANKEM**, **DHEERAJ SHETTY**, **PRAJAY MADHAVAN**, and **VICTOR LIN**. **SANKAR DEBNATH** was the system architect of the overall solution.

This team addressed all aspects of the CC3200Smart Plug development, from the conception, design, architecture, implementation, manufacturing and tests.