NXP Semiconductors
User's Guide

Document Number: WCT100XAV42LIBUG
Rev. 1, 07/2019

WCT1001A/WCT1003A V4.2 Transmitter Library

1. Introduction

This document describes the API of the WCT1001A/WCT1003A Wireless Charging Transmitter (WCT) library. The library enables you to evaluate the wireless charging Qi/PMA solution in custom applications easily.

This document describes the software features and enables you to develop custom applications based on the WCT1001A/WCT1003A transmitter (TX) library.

Contents

2. Overview

2.1. WCT software layers

The WCT library software layers are:

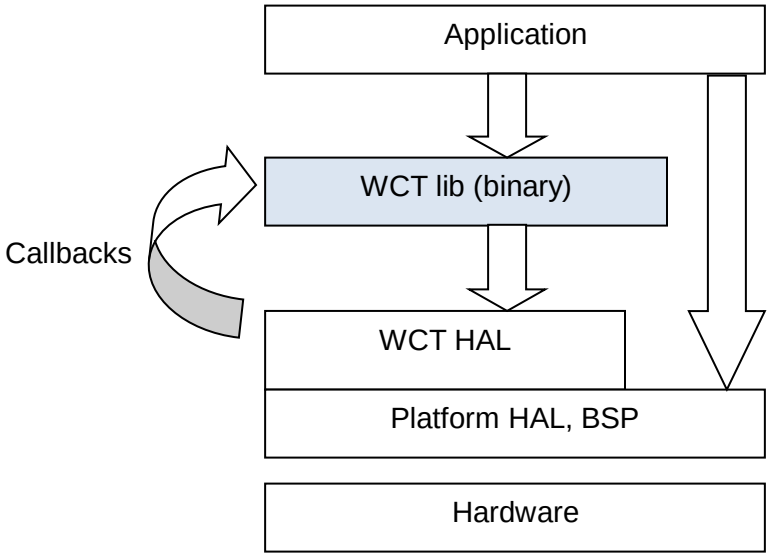

Figure 1. WCT library software layers

The WCT library is provided as a binary library, while the application and the Board Support Package (BSP) are in the source format.

The main modules of the WCT library include:

- WCT Qi/PMA state machine.
- Coil selection.
- Qi/PMA communication decoder.
- PID power-transfer control.
- Foreign Object Detection (FOD).

The WCT library API and the WCT Hardware Abstraction Layer (HAL) API are shown in the source format with these main functions:

- WCT library API:
  - Library version retrieval.
  - Library initialization.
  - Library main entry function.
  - Callbacks such as the tick timer callback and the Qi/PMA communication interrupt callback.
- WCT HAL API:
  - Coil-related HAL.

- Timer-related HAL.
- Voltage and current sensing HAL.
- Enable/disable interrupt HAL.

2.2. WCT software dynamics

There are two types of the A13 solution with different communication demodulations: Digital Demodulation (DDM) and Analog Demodulation (ADM).

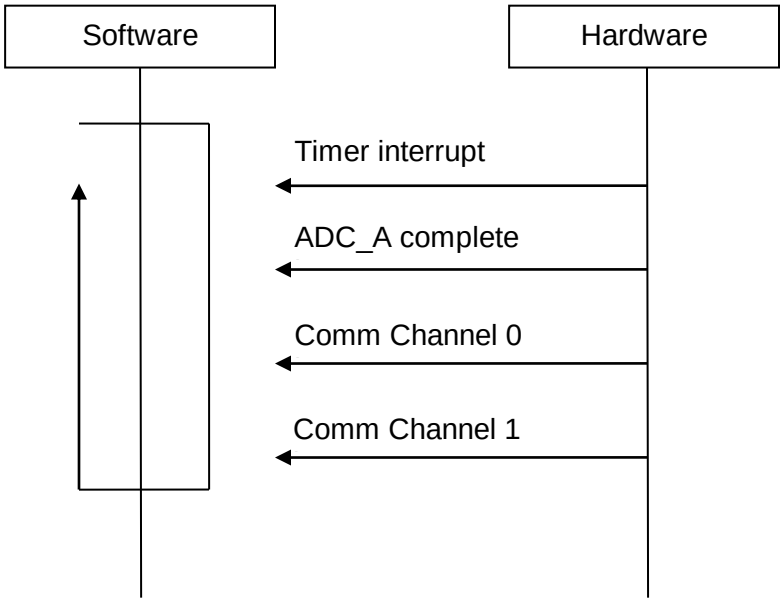

Figure 2. ADM demodulation

For the ADM demodulation:

- The main loop performs all the Qi/PMA functions, such as the state machine, the coil selection, the PID tuning, and the communication decoding.
- The timer interrupt is 200 µs. It triggers the ADC conversion and communication decoding. The voltage/current ADC sampling period is 200 µs.

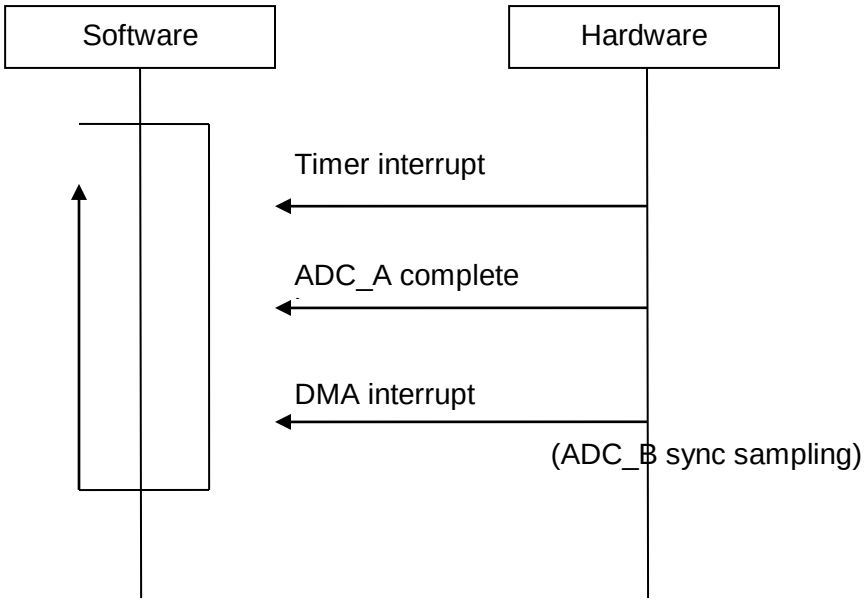

Figure 3. DDM demodulation

For the DDM demodulation:

- ADC_B is used to sample the coil current signal which is synchronized with the PWM frequency. This signal is used for the DDM.
- When a block (128 samples) of coil current data is saved, a DMA interrupt is triggered to let the software process it in a batch for the communication decoder.
- ADC_A is used to sample the input voltage (or current).

3. WCT library API

3.1. WCT charging mode

The WCT charging modes are indicated by the charge_mode variable in the WCTLIB_CFG_SWITCH structure.

Table 1. WCT charging mode

Definition		Description
#define CHARGE_MODE_DUAL_MODE	(0x03u)	WPC and PMA dual mode
#define CHARGE_MODE_WPC_ONLY	(0x02u)	WPC only mode
#define CHARGE_MODE_PMA_ONLY	(0x01u)	PMA only mode
#define CHARGE_MODE_RESERVED	(0x00u)	Reserved

3.2. WCT status and errors

3.2.1. WCT status

The WCT status is indicated by the TxChargingStatus variable in the RUNTIME_PARAMS structure.

Table 2. WCT status

Definition	Description
TX_ERROR_HALT	TX is halted due to error, NVM error, or chip error.
TX_APP_HALT	TX is halted in application.
TX_OBJECT_DETECTION	TX is detecting existence of the receiver (RX).
TX_COIL_SELECTION	TX is doing coil selection. The first round digital ping.
TX_COIL_SELECTION_CFM	TX is confirming the best coil. The second round digital ping.
TX_DIGITAL_PING	TX has found the best coil. The third round digital ping.
TX_IDENTIFICATION	TX is identifying the RX.
TX_CONFIGURATION	TX is collecting the configuration information about the RX.
TX_POWER_TRANSFER	TX is transferring power to the RX.
TX_RECHARGE_RETRY	TX or RX fault occurs, including FOD, APP error, EPT (Qi), and EOP (PMA).

3.2.2. WCT errors

The WCT errors are indicated by the TxChargingErrors variable in the RUNTIME_PARAMS structure.

Table 3. WCT errors

Definition	Description
TX_SUCCESS	TX is normal, no error occurs.
TX_CHIP_ERROR	Chip check error.
TX_NVM_ERROR	NVM check error.
TX_DRV_ERROR	TX driver is abnormal.
TX_FOD_ERROR	FOD occurs.
TX_APP_ERROR	APP error occurs.

3.2.3. WCR status

The WCR status is indicated by the RxChargingStatus variable in the RUNTIME_PARAMS structure.

Table 4. WCR status

Definition	Description
RX_NONE	RX is not on TX surface.
RX_PREPARE_CHARGE	RX is found, but is not in power-transfer mode yet.
RX_CHARGING	RX is in power-transfer mode, charging normally.
RX_CHARGED	Charge complete.
RX_UNDEFINE	When the TX FOD or App errors occur, or the charge-waiting time is over.
RX_FAULT	RX fault occurs.

3.2.4. WCR errors

The WCR errors are indicated by the RxChargingErrors variable in the RUNTIME_PARAMS structure.

Table 5. WCR errors

Definition	Description
RX_SUCCESS	RX is normal and no error occurs.
RX_WPC_EPT_UNKNOWN	WPC EPT unknown.
RX_WPC_EPT_INTERNAL_FAULT	WPC EPT internal fault.
RX_WPC_EPT_OVER_TEMP	WPC EPT over temperature.
RX_WPC_EPT_OVER_VOLT	WPC EPT over voltage.
RX_WPC_EPT_OVER_CURRENT	WPC EPT over current.
RX_WPC_EPT_BATTERY_FAILURE	WPC EPT battery failure.
RX_WPC_EPT_NO_RESPONSE	WPC EPT no response.
RX_WPC_EPT_RESERVED	WPC EPT reserved.
RX_WPC_PACKET_INCOMPATIBLE	WPC RX sends incompatible packets to TX.
RX_WPC_PACKET_RCVPWR_TIMEOUT	WPC Received Power Packet timeout.
RX_PMA_IDENTIFICATION_FAILURE	PMA RX identification fails.
RX_PMA_EOC_SYMBOLS	PMA RX sends EOC symbols.
RX_PMA_EOP_NO_LOAD	PMA EOP no load.
RX_PMA_EOP_RX_HOST	PMA EOP host request.
RX_PMA_EOP_INCOMPATIBLE_CLASS	PMA EOP incompatible power class.
RX_PMA_EOP_OVER_TEMP	PMA EOP over temperature.
RX_PMA_EOP_OVER_VOLT	PMA EOP over voltage.
RX_PMA_EOP_OVER_CURRENT	PMA EOP over current.
RX_PMA_EOP_OVER_DEC	PMA EOP over decrement.
RX_PMA_EOP_ALT_SUPPLY	PMA EOP alternate supply connected.
RX_PMA_EOP_INTERNAL_FAULT	PMA EOP internal fault.
RX_PMA_EOP_STABLE_FAILURE	PMA EOP voltage-stabilization error.
RX_PMA_EOP_COMM_ERROR	PMA EOP communication error.
RX_PMA_EOP_RECONFIGURE	PMA EOP reconfiguration.
RX_PMA_EOP_TBD	PMA EOP TBD.
RX_PMA_OVER_DEC	PMA over decrement.

3.3. WCT Library API functions

3.3.1. WCT_GetLibVer

Prototype:

```
WORD WCT_GetLibVer( void );
```

Description:

- Gets the WCT library version.

Return:

- WORD type with value format of x.y.z: 4-bit x, 4-bit y, and 8-bit z. For example, 0x2203 means version 2.2.3.

3.3.2. WCT_Init

Prototype:

```
void WCT_Init( BOOL bNeedReloadNVM );
```

Description:

- WCT library initialization. It initializes from the NVM parameters and resets the WCT internal states.

Parameters:

- bNeedReloadNVM:
  - 1: Need to reload the NVM parameters.
  - 0: No need to reload the NVM parameters.

3.3.3. WCT_Run

Prototype:

```
WORD WCT_Run( WORD wTimePassedMs );
```

Description:

- The main entry function of the WCT library. Make sure that this function is called within a 1-ms interval to ensure the timing requirements of the Qi/PMA certification.

Parameters:

- wTimePassedMs: the time elapsed since the last call of this function.

3.3.4. WCT_Stop

Prototype:

```
void WCT_Stop(void);
```

Description:

- Stops the WCT state machine from the application. To restart the WCT machine, call `WCT_Init()`. See the demo application.

3.3.5. WCT_SetParams

Prototype:

```
void WCT_SetParams(WORD data, BYTE devid, BYTE type);
```

Description:

- Sets the WCT parameters in run-time.

Parameters:

- data: the value of the data member to be set.
- devid: device ID.
- type: data member type.

Table 6. WCT_SetParams parameters

Type	Description	Remark
WCT_PINGDURATION	Set the digital ping duration.	—
WCT_PINGINTERVAL	Set the digital ping interval between adjacent pings.	—
WCT_PINGINTERVALLOOP	Set the digital ping interval between adjacent loop pings.	—
WCT_SETNEWFREQ	Set the new operation frequency during charging.	—

3.3.6. WCT_TickTimerHandler

Prototype:

```
void WCT_TickTimerHandler( void );
```

Description:

- The library callback function of the 1-ms tick timer. This timer is precise because it is a hardware interrupt. There are WCT internal functions that have to run exactly in a 1-ms interval.

3.3.7. WCT_TMR3TimerHandler

Prototype:

```
void WCT_TMR3TimerHandler( void );
```

Description:

- The library callback function of the TMR3 timer. It is used for data modulation from the TX to the RX in the PMA mode.

3.3.8. WCT_WpcDDMDataAnalyze

Prototype:

```
void WCT_WpcDDMDataAnalyze(BYTE byDevice);
```

Description:

- The library callback function of a DMA interrupt for the DDM only. In the current A13, this function is called when 128 samples of coil current are collected.

Parameters:

- byDevice: device ID.

3.3.9. WCT_PmaDDMDataAnalyze

Prototype:

```
void WCT_PmaDDMDataAnalyze(BYTE byDevice);
```

Description:

- Similar to the above.

Parameters:

- byDevice: device ID.

3.3.10. DisplayUpdateLedStatus

Prototype:

```
void DisplayUpdateLedStatus( BYTE byLedChannel, DEVICE_STATE eDeviceState );
```

Description:

- This function is implemented in the application to visualize the new state of the charger.

Parameters:

- byLedChannel: channel ID.
- eDeviceState: new device state.

3.3.11. WCT_GetDeviceFODParam

Prototype:

```
void WCT_GetDeviceFODParam(WORD wManuID, WORD wDevIDH, WORD wDevIDL, WORD
*pwPowerLimit, BYTE *pbyTripNum, DWORD *pdwCessationTime)
```

Description:

- This function informs you to set the FOD parameters based on the device information.
- The device information includes the manufacturing ID and the device ID (high and low words).
- The FOD parameters include power limit, TripNum, and CessationTime. During FO detection, if the power loss exceeds powerlimit by TripNum count and lasts for CessationTime, the FOD protection starts.
- This function gives you the flexibility and possibility to set different FOD parameters to handle certain RXs.

Parameters:

- wManuID: manufacturing ID.
- wDevIDH: device ID high word.
- wDevIDL: device ID low word.

Return:

- pwPowerLimit: power limit threshold.
- pbyTripNum: count threshold.
- CessationTime: time threshold.

3.3.12. WCT_Keyfob_Signal_Active

Prototype:

```
BOOL WCT_Keyfob_Signal_Active(void);
```

Description:

- This function is implemented in the application to inform the WCT library whether the keyfob avoidance is active or not. If enabled, the WCT library switches to a different operation

frequency for a short period of time.

Return:

- Flag indicating whether the keyfob avoidance is active or not.

3.3.13. OnDataPacketReceived

Prototype:

```
void OnDataPacketReceived(BYTE devid, BYTE bySize, BYTE *pbyData);
```

Description:

- This function is a callback function implemented in the application. When the TX receives a WPC data packet, this function is called to enable the application to check the packet data.

Parameters:

- devid: device ID.
- bySize: data size.
- pbyData: data buffer pointer.

3.3.14. WCT_OnPMASymbolRecv

Prototype:

```
void WCT_OnPMASymbolRecv(BYTE devid, POWERMAT_RX_MESSAGE_TYPE bSymbolType);
```

Description:

- This function is a callback function implemented in the application. When the TX receives a PMA symbol, this function is called to enable the application to check the packet data.

Parameters:

- devid: device ID
- bSymbolType: PMA symbol type

3.3.15. WCT_OnPMAPacketRecv

Prototype:

```
void WCT_OnPMAPacketRecv(BYTE devid, BYTE bySize, BYTE *pbyData);
```

Description:

- This function is a callback function implemented in the application. When the TX receives a PMA ES packet, this function is called to enable the application to check the packet data.

Parameters:

- devid: device ID.
- bySize: data size.
- pbyData: data buffer pointer.

3.3.16. WCT_CoilSelectorControl

Prototype:

```
WORD WCT_CoilSelectorControl( BYTE byDeviceId, WORD state, void *pParam );
```

Description:

- This function is implemented in the application to configure the coil-related parameters. The coil combination is organized by the application. This table describes the configuration types:

Table 7. WCT_CoilSelectorControl description

Type	Description	Remark
WCT_COIL_SELECTOR_COILDISABLE	Disable the coil combination selected.	Remove the PWM signal on the selected coil port.
WCT_COIL_SELECTOR_COILENABLE	Enable the coil combination selected.	Apply the PWM signal on the selected coil port.
WCT_COIL_SELECTOR_FREQSET	Set the frequency of the coil combination selected.	If the input pointer pParam is 0, set the ping frequency. Otherwise, set the frequency to the pointer value. To tune the frequency in the fixed operation frequency solution (A13), modify the pointer value too.
WCT_COIL_SELECTOR_VOLTSET	Set the voltage of the coil combination selected.	If the input pointer pParam is 0, set the ping voltage. Otherwise, set the voltage to the pointer value.
WCT_COIL_SELECTOR_DUTYCYCLESET	Set the duty cycle of the coil combination selected.	If the input pointer pParam is 0, set the ping duty cycle. Otherwise, set the duty cycle to the pointer value.
WCT_COIL_SELECTOR_ACTIVECOIL_SET	Set the active coil ID of the coil combination selected.	—
WCT_COIL_SELECTOR_RESET	Reset the coil selection module.	—

Parameters:

- byDeviceId: device ID.
- state: configuration types defined in the table above.
- pParam: input pointer.

Return:

- WCT error codes defined in the data types section.

3.3.17. WCT_CoilSelectorCheck

Prototype:

```
BYTE WCT_CoilSelectorCheck( BYTE byDeviceId, BYTE bySignalStrength, BYTE
byFaultCoilIdIndex);
```

Description:

- This function is implemented in the application to control the coil selection in multiple-coil solutions.

- The return types that affect the state machine execution are as follows:

Table 8. WCT_CoilSelectorCheck description

Type	Description	Remark
WCT_COIL_SELECTOR_STATE_NEXT	Indicate that not all coil combinations are searched and there are still coil combinations left to try.	—
WCT_COIL_SELECTOR_STATE_FAULTCOIL	The coil selected is the faulty coil ID input.	For example, the faulty coil may have input power exceeding a threshold.
WCT_COIL_SELECTOR_STATE_FOUND	After two rounds of search, a coil combination is found.	—
WCT_COIL_SELECTOR_STATE_NOTFOUND	No RX is found during a round of coil combination search.	—
WCT_COIL_SELECTOR_STATE_RETRY	All coil combinations are searched for the first round and the signal strength is available, which indicates that an RX exists on the TX surface.	Run a second round because the RX may not be on the TX surface from the start of a search round. That means the coil combination may not be the optimal one.

Parameters:

- byDeviceId: device ID.
- bySignalStrength: signal strength returned during digital ping.
- byFaultCoilIdIndex: faulty coil index.

Return:

- The return types are defined in the table above.

3.3.18. SetReChargeTimeOnAbnormal

Prototype:

```
WORD SetReChargeTimeOnAbnormal(WORD wAbnormalType);
```

Description:

- This function is implemented in the application to set the hold time. During wireless charging, certain abnormal conditions may occur. The state machine waits for a while before it restarts charging.
- The use cases with a different hold time recommended are:

Table 9. SetReChargeTimeOnAbnormal description

Condition	Description	Hold time
EPT – Unknown	RX sends the “End Power Transfer” packet with reason code "Unknown".	0 (exits transfer state)
EPT – Charge Complete	RX sends the “End Power Transfer” packet with reason code “Charge complete”.	5 minutes
EPT – Internal Fault	RX sends the “End Power Transfer” packet with reason code “Internal Fault”.	Infinity
EPT – Over Temperature	RX sends the “End Power Transfer” packet with reason code “Over Temperature”.	5 minutes
EPT – Over Voltage	RX sends the “End Power Transfer” packet with reason code “Over Voltage”.	0 (exits transfer state)
EPT – Over Current	RX sends the “End Power Transfer” packet with reason code “Over Current”.	0 (exits transfer state)

Table 9. SetReChargeTimeOnAbnormal description

Condition	Description	Hold time
EPT – Battery Failure	RX sends the “End Power Transfer” packet with reason code “Battery Failure”.	Infinity
EPT – No Response	RX sends the “End Power Transfer” packet with reason code “No Response”.	0 (exits transfer state)
EPT – Reserved	RX sends the “End Power Transfer” reserved packet (0x09-0xFF).	0 (exits transfer state)
Received Power Packet timeout	TX does not receive the “Received Power Packet” and exceeds the time threshold.	5 minutes (configurable because already put in configuration data)
FOD error	Foreign object detected.	5 minutes (configurable because already put in configuration data)
Input power exceed threshold	The calculated input power is too high and exceeds the threshold (configurable because already put in configuration data).	5 minutes
APP error	APP error happens.	5 minutes

Parameters:

- wAbnormalType: different abnormal types. Check the header file for all types.

Return:

- Hold time set by the user.

3.3.19. SetReChargeTimeOnPmaAbnormal

Prototype:

```
WORD SetReChargeTimeOnPmaAbnormal(WORD wAbnormalType);
```

Description:

- This function is similar to the previous one, except that it is for the PMA.
- The use cases with a different hold time recommended are:

Table 10. SetReChargeTimeOnPMAAbnormal description

Condition	Description	Hold time
Identification failed	TX cannot get RX’s RXID.	5 minutes
EOC	RX sends EOC symbols to TX.	5 minutes
PMA FOD	A foreign object is detected.	5 minutes
Extended EOP	RX sends extended EOP packet to TX.	According to the packet received, the recharging time can be modified.

Parameters:

- wAbnormalType: Different abnormal types. Check the header file for all types.
- dwInternalCalc: Internal preset recharging time, such as the internal sleep time from the Extended EOP packet.

Return:

- Hold time set by the user.

3.3.20. WCT_OpStateDebugMode

Prototype:

```
BYTE WCT_OpStateDebugMode(BYTE* debugMode);
```

Description:

- This function is used for the WCT GUI debugging purposes. Customers should not use it.

3.3.21. WCT_CalcInputPower

Prototype:

```
void WCT_CalcInputPower( RUNTIME_PARAMS *pRunTimeParams , BYTE byOffset);
```

Description:

- This function is used to calculate the input power.

3.3.22. WCT_CalcPowerLoss

Prototype:

```
void WCT_CalcPowerLoss( RUNTIME_PARAMS *pRunTimeParams );
```

Description:

- This function is used to calculate the power loss.

3.3.23. WCT_AppErrChkDuringWCTON

Prototype:

```
BOOL WCT_AppErrChkDuringWCTON(void);
```

Description:

- Certain errors occur at the application side during power transfer. The library provides a mechanism for the application to use the internal function of recharging retry timeout for such errors. The library calls this function during power transfer to check whether application errors occur. If an application error occurs, the library enables the application to configure the recharging retry time using the SetReChargeTimeOnAbnormal() interface.

Return:

- Flag when the APP error occurs.

3.3.24. WCT_WpcModulateHandle

Prototype:

```
void WCT_WpcModulateHandle(BYTE byDevice);
```

Description:

- This function is the handler to send an FSK response to the RX during Samsung fast charging.

3.4. WCT library configurations

The WCT configurations are in these places:

- WCHTX_STATIC_CONFIG structure.
- PMA_PARAMS_CONST structure.
- Non-Volatile Memory (NVM) parameters.

3.4.1. WCT_STATIC_CONFIG structure

```
typedef struct
{
   BYTE bNumDevices;
   BYTE bCommChannelsPerDevice;
   BYTE byNumCoilsPerDevice;
   BYTE byCEPTimeOutMultiRatio;
   BYTE byNumFreqBreakpoints;                  // not applicable to A13
   BYTE byPowerLossSampleBufSize;
   WORD wSizeOfSystemParams;
   WORD wSizeOfOpParams;
   WORD wSizeOfCalParams;
   DWORD dwCommReferenceTimerMaximumCount;
   DWORD dwSystemFlashBaseAddress;
   WORD wtick_intrupt_timeus;
   BYTE byCommEdgeBufferSize;
   BYTE byReserved;
   WORD wComm_Reference_Timer_Resolution_ns;
   WCTLIB_CFG_SWITCH wctlib_cfg_switch;

#if ( WCT_POWERMAT_ENABLE_FSL == WCT_TRUE )
   DWORD dwPowermatInputTimerMaximumCount;
   WORD wPowermat_Input_Timer_Resolution_ns;//POWERMAT_INPUT_TIMER_RESOLUTION_NS
#endif
#if (WCT_DEBUG == WCT_TRUE)
   WCTDBG_CONFIG wctdbg_cfg;
#endif
    WORD wPingIntervalAdjCoilMs;
    FOD_CHK_FUNC pFodChkFunc;
}WCT_STATIC_CONFIG;
```

The following table lists the meaning of each data member:

Table 11. Meaning of each data member

Data member	Description
bNumDevices	Device number. In the current A13, 1 device by default.
bCommChannelsPerDevice	Communication channels. For ADM, 2 hardware channels. For DDM, 4 filtering paths.
byNumCoilsPerDevice	Number of coils per device. In the current A13, 3 coils by default.

Table 11. Meaning of each data member

Data member	Description
byCEPTimeOutMultiRatio	Ratio multiplied to the CEP timeout to improve robustness.
wSizeOfSystemParams	Size of the system parameter structure in bytes.
wSizeOfOpParams	Size of the operation parameter structure in bytes.
wSizeOfCalParams	Size of the calibration parameters structure in bytes.
dwCommReferenceTimerMaximumCount	Maximum value for the count of communication reference timer. In the current A13, it is 65536 by default.
byNumFreqBreakpoints	Not applicable to A13.
byPowerLossSampleBufSize	Sample buffer size used for power loss calculation.
dwSystemFlashBaseAddress	Flash base address used to store NVM data.
wtick_intrupt_timeus	Tick interrupt time in µs.
byCommEdgeBufferSize	Internal usage for DDM.
byReserved	Reserved for aligning.
wComm_Reference_Timer_Resolution_ns	Communication reference timer resolution in ns.
wctlib_cfg_switch	Currently has 2 bits enabled: charge_mode: sets different working modes of the WCT library. See the WCT Charge Mode flag to enable the Freescale Powermat. pwrlossdetection: flag to enable FOD detection. libprotect_enable: flag to enable library internal protection, including input power protection and coil current protection. quickremove_enable: flag to enable quick RX removal detection. fastcharging_enable: flag to enable Samsung fast charging mode. fastchargingfod_enable: flag to enable FOD during Samsung fast charging mode.
dwPowermatInputTimerMaximumCount	Maximum count for Powermat input timer. For a 16-bit register, it is usually 65536.
wPowermat_Input_Timer_Resolution_ns	Powermat input timer resolution in ns.
wctdbg_cfg	Debug switches: generaldbg: prints general debug information. statemachinedbg: prints state machine states (internal states). commpacketdbg: prints communication data. piddbg: prints PID control information. Pmadbg: prints PMA-related information
wPingIntervalAdjCoilMs	Digital ping interval between adjacent pings.
pFodChkFunc	The API for the application to implement the FOD checking function. If it is NULL, the TX uses the internal default function.

3.4.2. PMA_PARAMS_CONST structure

The PMA (or AirFuel Inductive standard) has quit the wireless charging field and NXP does not support the PMA anymore.

3.4.3. NVM parameters

Table 12 describes the configurations in the NVM for the WCT100X library.

Table 12. NVM parameters

Name	Description	Remark
wPwmDeadTimeNs	Defines the dead time that is used for the PWM outputs for the FET driver.	nanosecond
wKeyfobAvoidanceDurationMs	Defines the amount of time when the unit operates at the Keyfob Avoidance Frequency after being triggered by the IO control signal. This value is ignored if the Keyfob Avoidance Duration Based on the IO parameter is TRUE.	millisecond
byKeyfobAvoidanceDurationBasedOnIo	If TRUE, the duration of the keyfob avoidance is determined by the state of the IO. If FALSE, the duration of the keyfob avoidance is determined by wKeyfobAvoidanceDurationMs.	—
byKeyfobAvoidanceDisableCoil	If TRUE, the coil is disabled while the keyfob detection is active. If FALSE, the frequency hopping keyfob avoidance strategy is used.	—
PowerControl	Bit 0: TRUE for frequency control. Bit 1: TRUE for rail voltage control. Bit 2–Bit 11: bits for enabling coils, Bit 2 for coil 0, and so on. Bit 12–Bit 15: bits for enabling devices, Bit 12 for device 0, and so on.	—
WpcProtections	Bit 0: TRUE to enable shutdown if the RX version is not compliant (not used). Bit 1: TRUE to enable exiting the power transfer state if a power packet is not received within the timeout. Bit 2: TRUE to disable analog ping.	—
dwPingFrequency	Coil operation frequency during ping stage.	105-115K
wPingDutyCycle	Duty cycle during ping stage.	In percentage, default 50
wPingPulseDurationTimeMs	Time duration to apply power signal during ping stage.	millisecond
wPingIntervalMs	Time interval between adjacent pings (between search loop).	—
dwAnalogPingFrequency	Coil operation frequency during analog ping stage.	—
wAnalogPingMinCoilCurrentThreshold	Feedback coil current threshold during analog ping stage. If less than this value, a hardware fault is triggered, which shuts down the device.	—
wAnalogPingCoilCurrentThreshold	Threshold above which a device is detected.	—
byAnalogPingDutyCycle	Duty cycle during analog ping stage.	Default 50
byAnalogPingPulseDuration	Analog ping time duration in number of cycles.	In number of cycles
byAnalogPingAdcSampleTime	Time length when ADC samples the coil current regarding the start of pulse in number of cycles.	In number of cycles
byDigitalPingRetryIntervalSeconds	Time interval for a second round of digital ping for device detection.	second
wOverCurrentThreshold	Overcurrent threshold.	mA
dwInputPowerThreshold	Maximum allowable input power.	mW
dwMinFreq	Minimum value of the operation frequency.	105-115K
dwMaxFreq	Maximum value of the operation frequency.	105-115K
dwKeyfobAvoidanceFreq	Operation frequency when Keyfob Avoidance is active.	—
wIntegralUpdateInterval	Time constant for integrator update rate.	millisecond, default 5
wDerivativeUpdateInterval	Time constant for derivative update rate.	millisecond, default 5

Table 12. NVM parameters

Name	Description	Remark
iIntegralUpperLimit	Maximum allowable value for the integral term of the PID control signal.	—
iIntegralLowerLimit	Minimum allowable value for the integral term of the PID control signal.	—
iPidUpperLimit	Maximum allowable value for the PID control signal.	—
iPidLowerLimit	Minimum allowable value for the PID control signal.	—
wPidScaleFactor	Scale factor for the PID control signal to generate the real control variable.	—
byDelayTimeMs	Time length between a control error packet received and the start of the PID tuning.	millisecond, defined in Qi specifications
byActiveTimeMs	Time while the PID tuning is allowed.	millisecond, defined in Qi specifications
bySettleTimeMs	Time while the PID tuning stabilizes.	millisecond, defined in Qi specifications
byNumPidAdjustmentsPerActiveWindow	Maximum number of PID iterations within the Active Time window.	—
byMaxDutyCycle	Maximum duty cycle allowable.	In percentage
byMinDutyCycle	Minimum duty cycle allowable.	In percentage
byDCStep	Duty cycle step when converting the PID output to a real control variable.	In percentage
byDCPidScaleFactor	Scale factor for the PID control signal to generate a real control variable of a duty cycle.	—
byDCKp	Duty cycle control proportional gain.	—
byDCKi	Duty cycle control integral gain.	—
byDCKd	Duty cycle control derivative gain.	—
wMinRailVoltageMv	Minimum operating rail voltage.	—
wMaxRailVoltageMv	Maximum operating rail voltage.	—
wDefaultRailVoltageMv	Default operating rail voltage.	—
wRailStepMv	Rail voltage step when converting the PID output to a real control variable.	—
wRailPidScaleFactor	Scale factor for the PID control signal to generate the real control variable of rail voltage.	—
byRailKp	Rail voltage control proportional gain.	—
byRailKi	Rail voltage control integral gain.	—
byRailKd	Rail voltage control derivative gain.	—
dwPowerLossIndicationToPwrCessationMs	Time it takes to let the TX to remove the power if the FOD keeps failing.	millisecond
dwPowerLossFaultRetryTimeMs	Time it takes for the TX to retry power transfer after a FOD fault is detected.	—
wPowerLossBaseThreshold	Base value to calculate the FOD power loss threshold.	—
wPowerLossIncrementalThreshold	Incremental value to calculate the FOD power loss threshold. Formula to calculate the FOD Threshold: FOD Base Threshold + (FOD Incremental Threshold * Bin#) In the current A13 solution, only the FOD Base Threshold is used.	—
byNumFodTripsToIndication	Number of consecutive threshold violations required to trigger an FOD indication.	—
byDefaultWindowOffset	Default FOD time window to support former legacy RXs older than v1.1.	—
wMinRailVoltageMv	This parameter is used for calibration and it is the minimum rail voltage the hardware is capable of producing.	—

Table 12. NVM parameters

Name	Description	Remark
wMaxRailVoltageMv	This parameter is used for calibration, and it is the maximum rail voltage the hardware is capable of producing.	—
sdwRailVoltageSlope	Defines the rail voltage normalized calibration slope.	—
sdwRailVoltageOffset	Defines the rail voltage normalized calibration offset.	—
sdwInputCurrentSlope	Defines the input current normalized calibration slope.	—
sdwInputCurrentOffset	Defines the input current normalized calibration offset.	—
wRailVoltageNorm	Defines the normalization factor used in the rail voltage normalized calibration.	—
wInputCurrentNorm	Defines the normalization factor used in the input current normalized calibration.	—
wInputVoltageCalibration	Indicates the calibration error for the ADC reading of the Input Voltage. For example, a value of 77 % (translated to a parameter value of 25231) indicates that the actual value of the Input Voltage is 77 % of the reported ADC value for the system.	—
wInputCurrentCalibration	Indicates the calibration error for the ADC reading of the Input Current. For example, a value of 77 % (translated to a parameter value of 25231) indicates that the actual value of the Input Current is 77 % of the reported ADC value for the system.	—
wCoilCurrentCalibration	Indicates the calibration error for the ADC reading of the Coil Current. A value of 77 % (translated to a parameter value of 25231) indicates that the actual value of the Coil Current is 77 % of the reported ADC value for the system.	—
wCoilCurrentDiodeDrop	Defines the nominal voltage drop of the diode used in the Coil Current peak detection circuitry.	mV
swQuadCoefficient	During characterization calibration, quadratic coefficient value to estimate the TX power losses (one for each coil).	Auto generated by calibration process
wQuadExponent	During characterization calibration, quadratic coefficient exponent value to estimate the TX power losses (one for each coil).	Auto generated by calibration process
swLinearCoefficient	During characterization calibration, linear coefficient value to estimate the TX power losses (one for each coil).	Auto generated by calibration process
wLinearExponent	During characterization calibration, linear coefficient exponent value to estimate the TX power losses (one for each coil).	Auto generated by calibration process
swConstantCoefficient	During characterization calibration, constant coefficient value to estimate the TX power losses (one for each coil).	Auto generated by calibration process
swPowerLossCalibrationOffset	During characterization calibration, this parameter accounts for potential errors that could result in a negative power loss calculation (one for each coil).	—
swQuadCoefficient	During normalization calibration, quadratic coefficient exponent value to estimate the redundant error after the characterization calibration (one for each coil).	Auto generated by calibration process
wQuadExponent	During normalization calibration, quadratic coefficient exponent value to estimate the redundant error after the characterization calibration (one for each coil).	Auto generated by calibration process

Table 12. NVM parameters

Name	Description	Remark
swLinearCoefficient	During normalization calibration, quadratic coefficient exponent value to estimate the redundant error after the characterization calibration (one for each coil).	Auto generated by calibration process
wLinearExponent	During normalization calibration, quadratic coefficient exponent value to estimate the redundant error after the characterization calibration (one for each coil).	Auto generated by calibration process
swConstantCoefficient	During normalization calibration, quadratic coefficient exponent value to estimate the redundant error after the characterization calibration (one for each coil).	Auto generated by calibration process

4. WCT HAL API

4.1. WCT Timer HAL API

4.1.1. WCT_TimerDelayMs

Prototype:

```
void WCT_TimerDelayMs( WORD wNumMsToDelay );
```

Description:

- This function delays the code execution for a certain amount of time in milliseconds.

Parameters:

- wNumMsToDelay: amount of time in ms.

4.1.2. TimerGetTimerTicks

Prototype:

```
WORD TimerGetTimerTicks( void );
```

Description:

- This function is implemented in the HAL layer to get the system timer ticks in milliseconds. Note that the value has a roll-back use case, and the roll-back count in A13 is 65536.

Return:

- System time ticks.

4.1.3. WCT_GetCommRefClockCount

Prototype:

```
WORD WCT_GetCommRefClockCount( void );
```

Description:

- This function is implemented in the HAL layer to get the communication reference clock count.

Return:

- Reference clock count.

4.2. WCT Coil HAL API

4.2.1. WCT_CoilInit

Prototype:

```
void WCT_CoilInit( void );
```

Description:

- This function is implemented in the HAL layer to initialize the coils, including the PWM channels.

4.2.2. WCT_CoilFreqSet

Prototype:

```
BYTE WCT_CoilFreqSet( BYTE byCoilId, DWORD dwFreqValue );
```

Description:

- This function is implemented in the HAL layer to set the frequency of a defined coil.

Parameters:

- byCoilId: coil ID.
- dwFreqValue: frequency to be set.

Return:

- RET_VALUE_GOOD.
- RET_VALUE_INVALID_PARAM.

4.2.3. WCT_CoilDutyCycleSet

Prototype:

```
BYTE WCT_CoilDutyCycleSet( BYTE byCoilId, WORD wDutyCyclePercent );
```

Description:

- This function is implemented in the HAL layer to set the duty cycle of a defined coil.

Parameters:

- byCoilId: coil ID.
- wDutyCyclePercent: duty cycle to be set.

Return:

- RET_VALUE_GOOD.
- RET_VALUE_INVALID_PARAM.

4.2.4. WCT_CoilControl

Prototype:

```
BYTE WCT_CoilControl( BYTE byCoilId, BOOL bState );
```

Description:

- This function is implemented in the HAL layer to enable or disable a certain coil.

Parameters:

- byCoilId: coil ID.
- bState:
  - WCT_ENABLE: enable the selected coil.
  - WCT_DISABLE: disable the selected coil.

Return:

- RET_VALUE_GOOD.
- RET_VALUE_INVALID_PARAM.

4.2.5. coil_power_enable

Prototype:

```
void coil_power_enable(void);
```

Description:

- This function is implemented in the HAL layer to enable the power connected to the resonance circuit.

4.2.6. coil_power_disable

Prototype:

```
void coil_power_disable(void);
```

Description:

- This function is implemented in the HAL layer to disable the power connected to the resonance circuit.

4.2.7. WCT_CoilAnalogPing

Prototype:

```
WORD WCT_CoilAnalogPing( BYTE byCoilId, BYTE byPulseDuration, BYTE byAdcSampleTime);
```

Description:

- This function is implemented in the HAL layer to perform an analog ping to detect if an object is placed on the charging surface.

Parameters:

- byCoilId: coil ID.
- byPulseDuration: pulse duration of analog ping.
- byAdcSampleTime: time value after which ADC is enabled.

Return:

- Sampled current value.

4.2.8. WCT_CoilDischarge

Prototype:

```
BYTE WCT_CoilDischarge( BYTE byCoilId, BOOL bState );
```

Description:

- This function is implemented in the HAL layer to release the power in the resonance circuit.

Parameters:

- byCoilId: coil ID. In the current A13, the 3 coils use the same discharging path.
- bState:
  - WCT_ENABLE: enable discharging.
  - WCT_DISABLE: disable discharging.

Return:

- RET_VALUE_GOOD.
- RET_VALUE_INVALID_PARAM.

4.2.9. WCT_CoilCheckHardwareOvercurrentStatus

Prototype:

```
BOOL WCT_CoilCheckHardwareOvercurrentStatus( BYTE byCoilId );
```

Description:

- Not used in the current A13 solution.

4.2.10. WCT_FSKSetFreq

Prototype:

```
void WCT_FSKSetFreq(SWORD adj);
```

Description:

- This function is used to set the FSK frequency when the TX responds to the RX data packet during Samsung fast charging mode.

Parameters:

- adj: Period register value to modulate the frequency.

4.3. WCT Analog IO HAL API

4.3.1. WCT_AnalogIoInit

Prototype:

```
void WCT_AnalogIoInit( RUNTIME_PARAMS* pRunTimeParams );
```

Description:

- This function is implemented in the HAL layer to initialize parameters for the Analog I/O, such as calibration constants.

Parameters:

- pRunTimeParams: run-time parameter pointer.

4.3.2. WCT_AnalogIoGetPowerSupplyVoltage

Prototype:

```
WORD WCT_AnalogIoGetPowerSupplyVoltage( VOLTAGE_REPORT eVoltageReport );
```

Description:

- This function is implemented in the HAL layer to get the power supply voltage value. In the current A13, this function is not used.

Parameters:

- eVoltageReport: voltage value type.

Return:

- Voltage value.

4.3.3. WCT_AnalogIoGetCoilCurrent

Prototype:

```
WORD WCT_AnalogIoGetCoilCurrent( BYTE byCoilId, CURRENT_REPORT eCurrentReport );
```

Description:

- This function is implemented in the HAL layer to get the coil current value.

Parameters:

- byCoilId: coil ID.
- eCurrentReport: current value type. In the current A13, the count is used.

Return:

- Coil current value.

4.3.4. WCT_AnalogIoGetInputCurrent

Prototype:

```
WORD WCT_AnalogIoGetInputCurrent( BYTE byCoilId, CURRENT_REPORT eCurrentReport );
```

Description:

- This function is implemented in the HAL layer to get the input current value.

Parameters:

- byCoilId: coil ID.
- eCurrentReport: current value type. In the current A13, the count is used.

Return:

- Input current value.

4.3.5. WCT_EnableAdcConversions

Prototype:

```
void WCT_EnableAdcConversions( void );
```

Description:

- This function is implemented in the HAL layer to trigger the ADC conversion.

4.3.6. WCT_AnalogIoGetRailVoltage

Prototype:

```
WORD WCT_AnalogIoGetRailVoltage( BYTE byDeviceId, VOLTAGE_REPORT eVoltageReport);
```

Description:

- This function is implemented in the HAL layer to get the rail voltage value.

Parameters:

- byDeviceId: device ID.
- eVoltageReport: voltage value type. In the current A13, the count is used.

Return:

- Rail voltage value.

4.3.7. WCT_AnalogIoSetRailVoltage

Prototype:

```
BYTE WCT_AnalogIoSetRailVoltage( BYTE byDeviceId, WORD wRailVoltageMv );
```

Description:

- This function is implemented in the HAL layer to set the rail voltage value.

Parameters:

- byDeviceId: device ID.
- wRailVoltageMv: Rail voltage in mV.

Return:

- RET_VALUE_GOOD.
- RET_VALUE_BAD.
- RET_VALUE_INVALID_PARAM.

4.4. WCT DDM HAL API

4.4.1. DDM_Start

Prototype:

```
void DDM_Start(BYTE byDevice)
```

Description:

- Enables the digital demodulation by correctly configuring and enabling the DMA channel.

Parameters:

- byDevice: device ID.

4.4.2. DDM_Stop

Prototype:

```
void DDM_Stop(BYTE byDevice)
```

Description:

- Stops the digital demodulation by stopping the DMA channel.

Parameters:

- byDevice: device ID.

4.4.3. DDM_SetBestTriggerPos

Prototype:

```
WORD DDM_SetBestTriggerPos(BYTE byDevice)
```

Description:

- Sets the best PWM trigger position to locate the valley point of the coil current signal.

Parameters:

- byDevice: device ID.

Return:

- Coil current value removing offset.

4.4.4. PWM_SetTriggerPos

Prototype:

```
void PWM_SetTriggerPos(BYTE byDevice, BYTE byPos);
```

Description:

- Sets the PWM trigger pos offset.

Parameters:

- byDevice: device ID.
- byPos: position.

4.4.5. ADC_ManualConversion

Prototype:

```
WORD ADC_ManualConversion(BYTE byDevice, WORD wAvgCnt, BYTE byIsTrigByPwm);
```

Description:

- Gets the ADC value; the ADC sample can be triggered by a PWM or manually.

Parameters:

- byDevice: device ID.
- byAvgCnt: measure count.
- isTrigByPwm: PWM trigger flag. 0: manually; 1: PWM trigger.

Return:

- Coil current sample value.

4.5. WCT Power HAL API

4.5.1. WCT_CvtInputVolmvFrmCnt

Prototype:

```
WORD WCT_CvtInputVolmvFrmCnt(DWORD dwVoltageCounts);
```

Description:

- This function is implemented in the HAL layer to convert the count value to the input voltage in units of mV.

Parameters:

- dwVoltageCounts: voltage count value.

Return:

- Input voltage value in units of mV.

4.5.2. WCT_CvtInputCurmaFrmCnt

Prototype:

```
WORD WCT_CvtInputCurmaFrmCnt(DWORD dwCurrentCounts);
```

Description:

- This function is implemented in the HAL layer to convert the count value to the input current in units of mA.

Parameters:

- dwCurrentCounts: current count value.

Return:

- Input current value in units of mA.

4.5.3. WCT_CvtCoilCurmaFrmCnt

Prototype:

```
WORD WCT_CvtCoilCurmaFrmCnt(DWORD dwCurrentCounts);
```

Description:

- This function is implemented in the HAL layer to convert the count value to the coil current in units of mA.

Parameters:

- dwCurrentCounts: current count value.

Return:

- Coil current value in units of mA.

4.5.4. WCT_GetPowerLimitOffset

Prototype:

```
WORD WCT_GetPowerLimitOffset(WORD wManuID, WORD wDevIDH, WORD wDevIDL);
```

Description:

- This function is implemented in the application to set the power limit offset for the FOD threshold.

Parameters:

- None.

Return:

- Power limit offset.

5. Library occupied peripherals

Table 13 describes the library occupied peripherals.

Table 13. Library occupied peripherals

Function	Interface	Remark
PWM	PWM3	Drives the resonance circuit.
DAC	DACA	Used for rail voltage control.
ADC	ADCA	Used to sample low-frequency signals, such as input current and input voltage.
ADC	ADCB	Used to sample coil current signal for DDM.
Timer	TMRA2	One tick timer with a periodic interrupt.
DMA interrupt	TMRA0	Used to generate the DMA interrupt.
DMA	DMA Channel0	Used for DDM with ADC samples.
Counter	PIT0	High-resolution counter to measure the time duration of the communication.
GPIO	GPIO	ON/OFF signal input/output; for example, enable/disable each coil, enable/disable power input, and so on. • DCDC_EN: used to switch NCV3011 on/off, see ENABLE_VRAIL_DCDC. • Coil_DIS: used to discharge coil, see COIL_DISCHARGE_CONTROL_PORTS. • VBATSW_EN: used to switch VBATSW on/off, see coil_power_enable(). • COIL0_EN, COIL1_EN, COIL2_EN: used to enable/disable coil, see COIL_ENABLE_CONTROL_PORTS.
Misc	TMRA3	Used for PMA voltage sweeping and advertising.

6. Typical application

6.1. Demo application

See the demo application in the release package.

6.2. Dynamic timing analysis

This section describes the WCT library dynamic timing analysis for your application's performance consideration.

The below data are measured with the chip running at 100 Mbit/s.

In the DDM solution, the coil current is sampled in the ADC_B and synced with the PWM frequency. When a block (128 samples) of coil current data is stored, a DMA interrupt is triggered to let the software process it in a batch for DDM filtering. The below time count uses 2560 ns as a time resolution.

- DDM filtering: 128 points to be processed together with the interrupt.
  Data time interval: 128 * 1/110K = 1164 µs.
  Processing counter value for WPC Qi: 72, corresponding time interval: 184 µs.
  Processing counter value for PMA: 109, corresponding time interval: 280 µs.

- CommCallback: convert from edge to byte.

  Trigger time interval: 1 ms (in tick timer interrupt).

  Processing counter value for WPC Qi: 40, corresponding time interval: 102 µs.

  Processing counter value for PMA: 2, corresponding time interval: 5 µs.
- ADC complete interrupt: Trigger time interval: 1 ms (in tick timer interrupt), can be omitted because the processing time is trivial.
- Main loop:
  - Most use cases: Processing counter value ~15, corresponding time interval 38 µs.
  - Rare use case: Processing counter value ~450, corresponding time interval 1152 µs. Due to the additional DDM function to re-sync the sampling point when receiving a data packet (which may use 90 points), corresponding to a delay of 818 µs (90 * 1/110K).

Figure 4 shows the time slot of the WPC Qi/PMA DDM software processing:

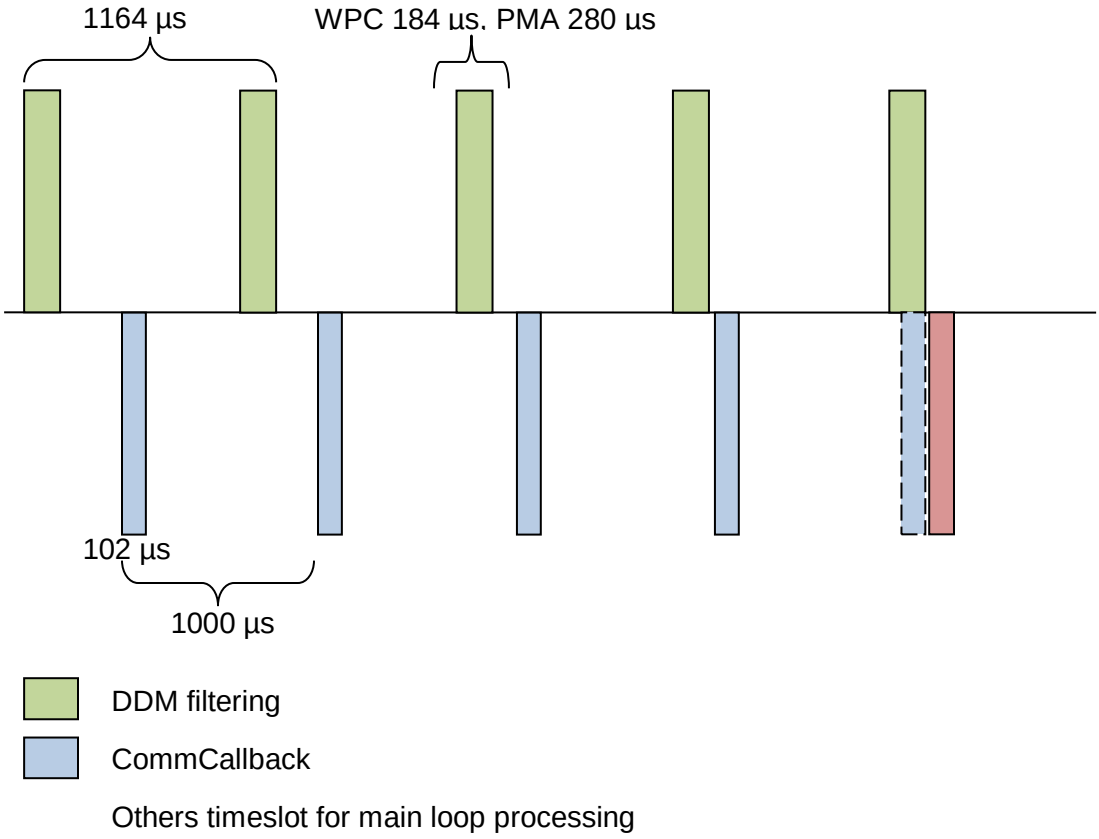

Figure 4. Time slot of WPC Qi/PMA software processing

7. New features of the library

Compared with the Version 4.0 release, the library in Version 4.2 has these features:

- Supports Samsung fast charging mode.
- Improves the power loss calculation robustness by confirming the data packet receipt.
- Fixes the DDM false decoding issue.

8. Revision history

The following table provides the revision history.

Table 14. Revision history

Revision number	Date	Substantive changes
0	05/2016	Initial release
1	07/2019	Update according to WCT library v4.2 changes.

How to Reach Us:

Home Page:
www.nxp.com

Web Support:
www.nxp.com/support

Document Number: WCT100XAV42LIBUG
Rev. 1
07/2019