台 揚 科 技 股 份 有 限 公 司

MICROELECTRONICS TECHNOLOGY INC.

ISO 9001 Certified

MTI MSx10 Development Kit

User’s Guide

Version 1.2a

IMPINJ CONNECTED

Contents

1 OVERVIEW

This document describes getting started with revision 0.03 of the MTI MSx10 development board and the version 1.02.00 MTI MSx10 SDK. For more details on the reference design hardware, see the MTI MSx10 Development Board Application Note.

2 KIT CONTENTS

The MTI MSx10 development kit contains the following:

1. MTI MSx10 Development Board revision 0.01
2. Raspberry Pi 4 Model B Rev 1.2/1.4 Single Board Computer (2GB-9003 or 4GB-9004)
3. Micro SD Card 16 GB with Raspberry Pi OS installed
4. Standoffs
5. Raspberry Pi HAT header extension (Pins 2 & 4 removed to isolate power)
6. 5V barrel jack power supply
7. 18W USB C power supply

The kit does NOT contain:

1. RFID cables or antennas
2. USB keyboard and mouse
3. Micro HDMI cable or monitor
4. Ethernet cable
5. Tag sample

3 QUICK START STEPS

To read tags with the kit, follow the steps below. Reference Figure 1, Figure 2, Table 1, and Table 2 for hardware connections.

Note: These steps assume that the kit is in its' initial state as shipped from MTI. Any modifications to the electrical, mechanical, or software configuration may change the behavior of the kit.

1. Connect the combined reader hardware (MTI MSx10 development board + Raspberry Pi 4 with Micro SD card)
   a. Connect 5V power via the barrel jack connector J9
   b. Connect USB type C power connector to Raspberry Pi
   c. Connect an RFID antenna to the Antenna via the Ipex connector J5
   d. Connect the Raspberry Pi to a network or to a monitor + mouse + keyboard
      i. To connect to a network, connect the Raspberry Pi Ethernet connector to a computer using Cat-5 or equivalent
      ii. To connect to a monitor + mouse + keyboard, use the micro HDMI and USB ports on the Raspberry Pi
2. Log in to the Raspberry Pi, either locally or on the network
   a. Locally using a monitor, mouse, and keyboard, log in with the following credentials (for Raspberry Pi's configured by MTI):
      i. Username: **`pi`**
      ii. Password: **`rpi2020`**
   b. Via the network, SSH to the static IP address of Raspberry Pi, which is **`192.168.0.10`**. The host computer must be set in the same network domain as the Raspberry Pi. Remotely via SSH using a tool like MobaXterm, PuTTY, Tera Term, etc. The following is using command prompt to log in.
      i. Launch the **`Command Prompt`** in Windows
      ii. Command: **`ssh pi@192.168.0.10`**
      iii. Passwrod: **`rpi2020`**
3. Determine RF connectivity for targeted operating application
   a. Connect RF cables for low power with internal PA (default) or high power with external PA (compliance Impinj Ex10 development board)
   b. Connect RF cables for monostatic (default) or bistatic
4. Execute the example script
   a. Navigate to the **`msx10_c_dev_kit`** directory by typing:

      ```
      cd ~/msx10_bundle_v1.00.12/msx10_c_dev_kit/
      ```
   b. Build the SDK by typing:

      ```
      make clean && make
      ```
   c. Execute the example inventory script by typing:

      For low power use case with internal PA of TX RF cable connectivity:

```
./build/e710_ref_design/examples/inventory_internal_pa.bin
```

5. Tag reads should appear in the console, as the reader performs fixed Q inventory for the duration of the script

IMPORTANT:

The C host library must be used with the exact same version application image.
To obtain version information from the MTI MSx10 SiP, see the Get Version Information section.
To upload an application image to the MTI MSx10 SiP, see the Application Image Upload section.

These TX RF cable connections on the development board must match the internal or external PA hardware configuration of the calibration flash page in MTI MSx10 SiP. See the Calibration Information section to know the detailed information.

Figure 1 - MTI MSx10 Development Kit Hardware Connections

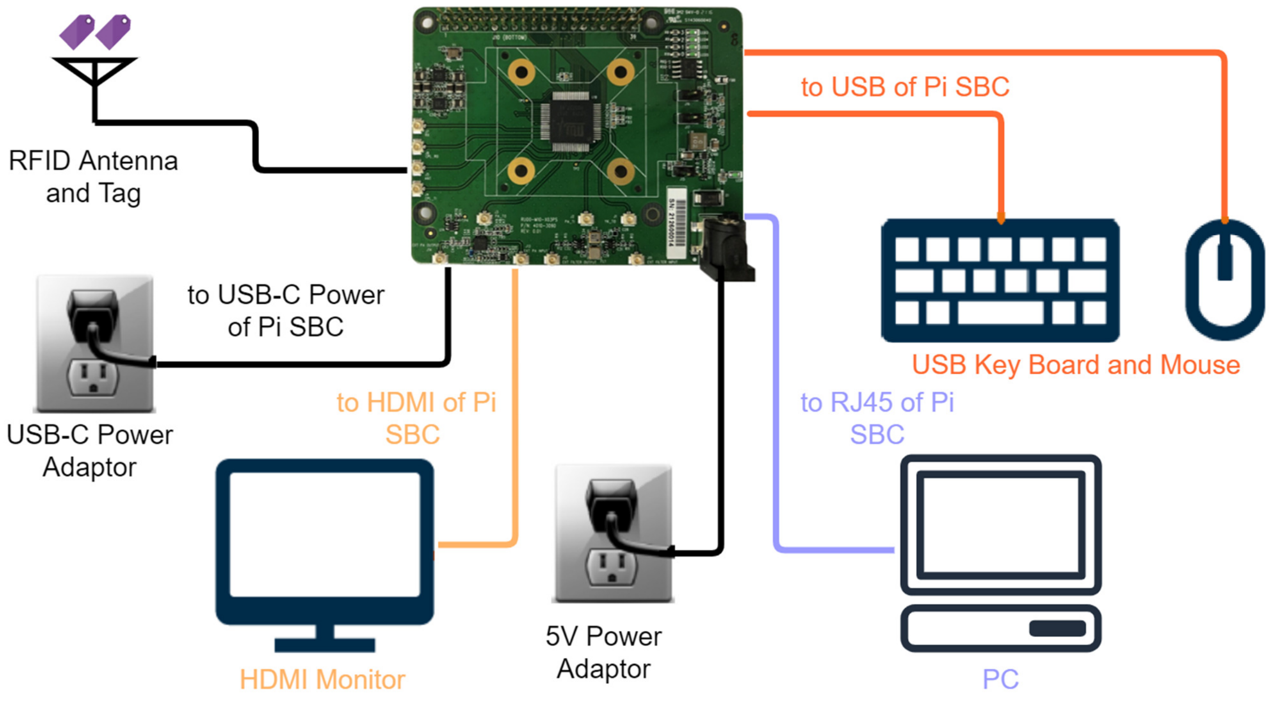
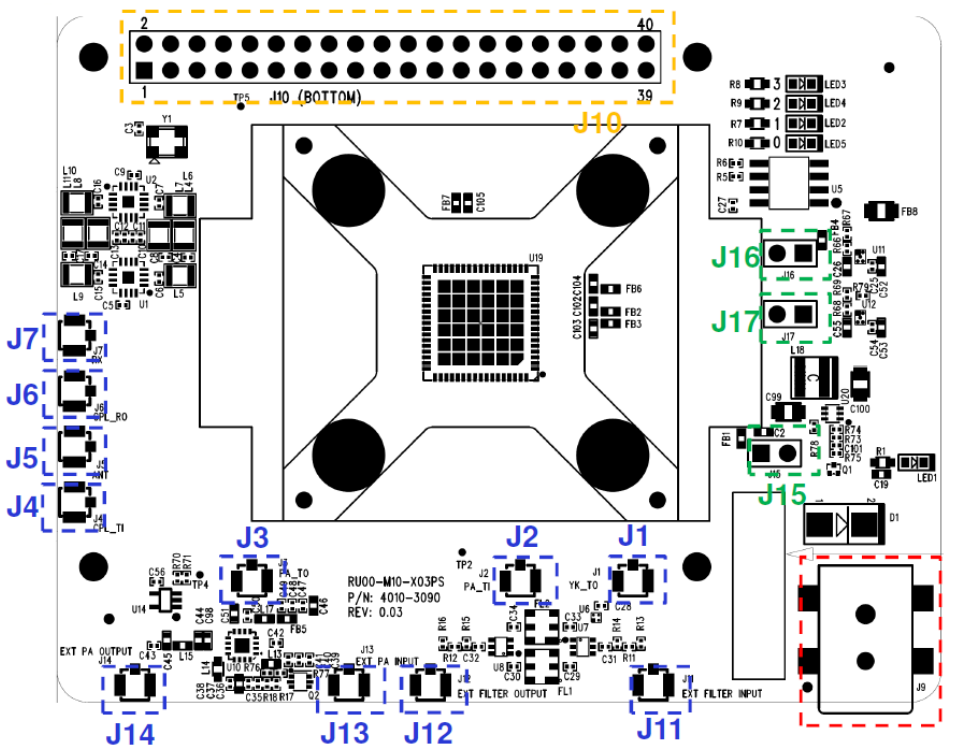


Figure 2 - MTI MSx10 Development Board Connectors



Power & Control :

J9	DC5V
J10	Raspberry Pi 40pin control inerface
J15	VDD_SIP3
J16	VDD_SIP1
J17	VDD_SIP2

RF Connectors :

J1	YK_TO
J2	PA_TI
J3	PA_TO
J4	CPL_TI
J5	ANT
J6	CPL_RO
J7	RX
J11	EXT FILTER INPUT
J12	EXT FILTER OUTPUT
J13	EXT PA INPUT
J14	EXT PA OUTPUT

Table 1 - TX RF Cable Connectivity

Use Case	TX RF Cable Connectivity	Description
Low Power (default)	J1 to J11	Buil-in Ex10 TX output ot external SAW filter input
	J12 to J2	External SAW filter output to internal PA input
	J3 to J4	Internal PA output to directional couple input
	J5 to antenna	SiP antenna port output to external antenna
High Power	J1 to J11	Built-in Ex10 TX output to external SAW filter input
	J12 to J13	External SAW filter output to external PA input
	J14 to J4	External PA output to directional coupler input
	J5 to antenna	SiP antenna port output to external antenna

Table 2 - RX RF Cable Connectivity

Use Case	RX RF Cable Connectivity	Description
Monostatic (default)	J6 to J7	Coupler reverse port to SiP RX
Bistatic	J6 to 50 ohm terminal	External SAW filter output to internal PA input
	J7 to RX antenna	SiP receiver port input to external antenna

4 MTI READER SIP SDK

4.1 Overview

The MTI MSx10 SDK contains specific configuration and example code. Its development is based on the Impinj Ex10 SDK which contains source code examples in C, host library source code in C and an Impinj Ex10 firmware binary called application image as shown in Figure 3.

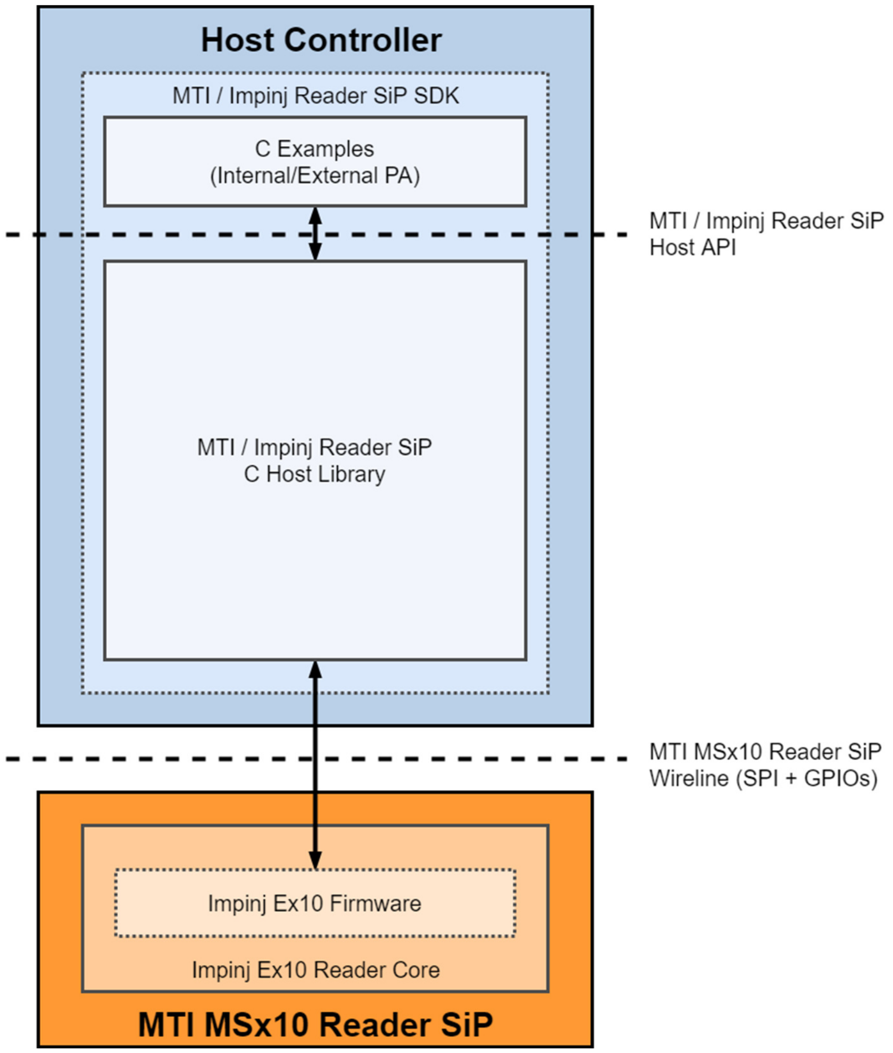
The host C library source, examples and board specific source can be found in the following directory.
`~/msx10_bundle_v1.02.00/msx10_c_dev_kit`

An application image can be found in the following directory.
`~/msx10_bundle_v1.02.00/ex10_app_image`

The API guide document can be found in the following directory. The API guide document explains how to use the provided host C library or reimplement it to suit the needs of a custom design.
`~/msx10_bundle_v1.02.00/ex10_docs`

The calibration data can be found in the following directory. See the Calibration Information section for detailed information.
`~/ex10_board_cal`

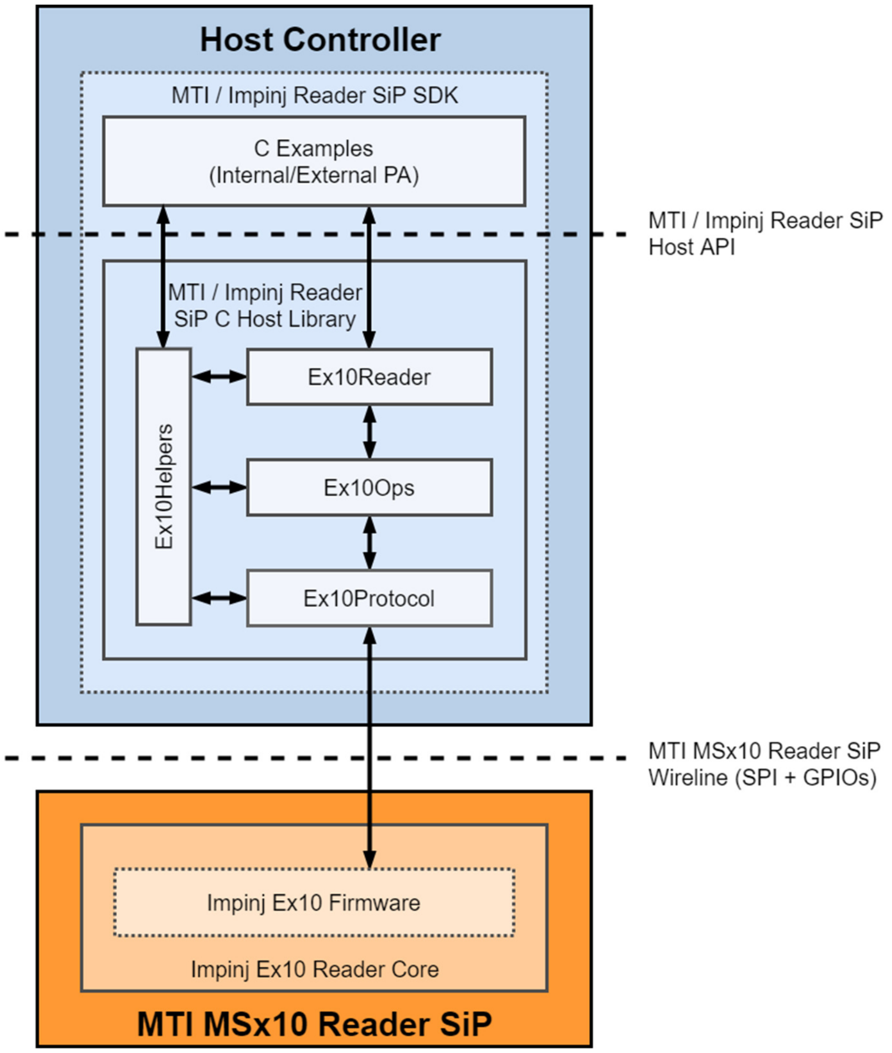
Figure 3 - MTI MSx10 SDK Overview


Host Controller
MTI / Impinj Reader SiP SDK
C Examples (Internal/External PA)
MTI / Impinj Reader SiP Host API
MTI / Impinj Reader SiP C Host Library
MTI MSx10 Reader SiP Wireline (SPI + GPIOs)
Impinj Ex10 Firmware
Impinj Ex10 Reader Core
MTI MSx10 Reader SiP

4.2 C Host Library

The host C library is made up of several C source files, implementing different aspects of the MTI reader SiP host. This host C library is used in the implementation of the provided C code examples. Within the host library source there are 3 layers of abstraction. The top layer, `Ex10Reader`, exposes an interface that implements highly integrated RAIN RFID functionality such as inventory and selecting tags. The middle layer, `Ex10Ops`, exposes an interface that abstracts MTI reader SiP modem operations. The lowest layer, `Ex10Protocol`, implements the SPI interface functionality that the host uses to communicate with the MTI reader SiP. This relationship is seen in Figure 4. The interface exposed by the `Ex10Reader` layer is called the MTI reader SiP host API and the interface exposed by the MTI reader SiP is called the MTI reader SiP wireline.

Figure 4 - MTI Reader SiP C Host Library Hierarchy



The MTI reader SiP host C library source can be found in the `~/msx10_bundle_v1.02.00/msx10_c_dev_kit/src/ex10_api` and `~/msx10_bundle_v1.02.00/msx10_c_dev_kit/include/ex10_api` directories with board specific implementations in the `~/msx10_bundle_v1.02.00/msx10_c_dev_kit/board` directory. This code is intended for use as-is, with modifications, or as a reference for developing an entirely new host library. Reference the provided source code license for more details. A changelog for each SDK release is available in the SDK documentation at the following location:
`~/msx10_bundle_v1.02.00/ex10_docs/yk_firmware/docs/yk_firmware_ex10_changelog.html`

~/msx10_bundle_v1.02.00/msx10_c_dev_kit/src/ex10_api/

- **aggregate_op_builder.c**
  - Implementation of the aggregate operation builder module.
- **application_registers.c**
  - Defines application register names and bit fields within a structure and provides access to those registers with a function.
- **board_init.c**
  - Functions for board level initialization to application and bootloader context.
- **command_transactor.c**
  - Implementation of protocol level read and write command transactions.
- **commands.c**
  - Contains implementations of MTI reader SiP commands.
- **crc16.c**
  - CRC16 function implementation.
- **event_fifo_printer.c**
  - Contains helper functions for printing EventFifo packet contents.
- **event_packet_parser.c**
  - Event FIFO packet parsing functions.
- **ex10_helpers.c**
  - Helper functions for various code examples.
- **ex10_lbt_helpers.c**
  - Helper functions for Listen before talk (LBT) code examples.
- **ex10_ops.c**
  - Implementation of MTI reader SiP operations with provided register settings.
- **ex10_power_modes.c**
  - Implementation of the reader SiP power modes interface.
- **ex10_protocol.c**
  - Implementation of the physical layer interface to the MTI reader SiP.
- **ex10_reader.c**
  - Implementation of the top-level API interface for the MTI reader SiP.
- **fifo_buffer_list.c**
  - Host EventFifo management functions.
- **gen2_commands.c**
  - Functions for encoding Gen2 command packets.
- **gen2_tx_command_manager.c**
  - Implementation of the Gen2v2 command sequence builder.
- **list_node.c**
  - Host EventFifo list management functions.
- **power_transactor.c**
  - Helper functions for powering the MTI reader SiP to various states.
- **print_data.c**
  - Helper functions for printing data to the terminal.
- **regions_table.c**

        - Functions and definitions related to regulatory region settings. Custom regional configurations can be defined here.
    - **`sjc_accessor.c`**
        - Functions for manipulating the SJC configuration.
    - **`trace.c`**
        - LTTng trace support.
    - **`version_info.c`**
        - Functions for obtaining version information from the MTI reader SiP.

4.3 C Host Examples

The MTI reader SiP SDK contains example code that demonstrates the use of the MTI reader SiP host API for the purpose of performing various Gen2 RAIN RFID commands. The example code is written in C and is targeted at the Raspberry Pi 4 host device. However, the examples (and host library) are intended to be portable to other host devices. The examples can also be used as a reference to write entirely new host application code.

The MTI reader SiP SDK contains utility example c code for testing purposes. Utilities may have at least one parameter, the host can execute a utility example binary with -h or --help to get the usage information.

4.3.1 Compiling the C SDK

The MTI reader SiP host C library and examples are distributed as source and need to be compiled before they can be executed. Upon compiling the source, multiple libraries and application binaries are produced in the directory listed below:

```
~/msx10_bundle_v1.02.00/msx10_c_dev_kit/build/e710_ref_design/
```

The following steps will demonstrate how the host library and examples can be compiled to run on the Raspberry Pi host.

The top level Makefile is in the following directory:

```
~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
```

Open a terminal on the Raspberry Pi and execute the following command to navigate to that directory:

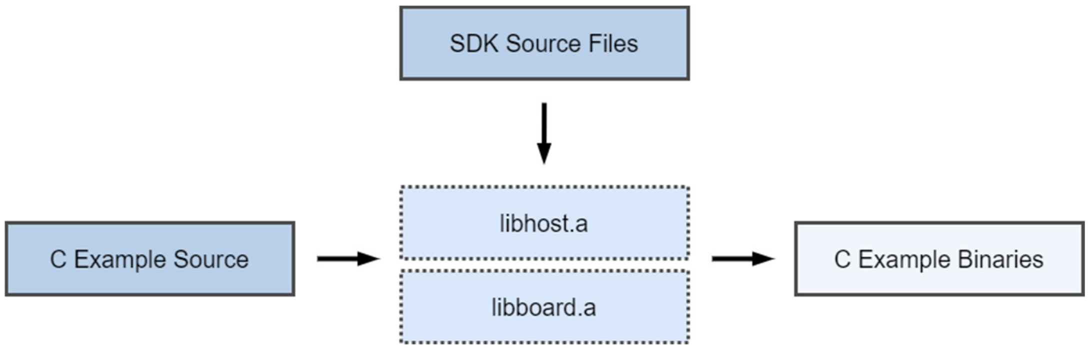
```
cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
```

Execute the following commands to compile the examples for the MTI MSx10 development kit:

```
make clean && make -j6
```

Static libraries libboard.a and libhost.a are created during the build process as shown in Figure 5. Libboard is a board specific static library while libhost is a hardware agnostic static library that supports API calls implemented by the provided MTI reader SiP SDK.

Figure 5 - Compiling the MTI Reader SiP SDK Source to Libraries and Binaries



Once the MTI reader SiP SDK and examples have been compiled, the libraries and C example binaries can be found in the following directory:

```
~/msx10_bundle_v1.02.00/msx10_c_dev_kit/build/e710_ref_design/
```

4.3.2 Get Version Information

The get version info example demonstrates how to obtain version information from the MTI MSx10 SiP. Executing the get version info example without arguments defaults to returning all version information. Passing multiple arguments requires each argument to be space delimited. See the corresponded examle file to find the accepted arguments.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the version info example binary:

   ```
   ./build/e710_ref_design/examples/get_version_info.bin
   ```

After executing these steps, the binary will run and output the following:

```
Application:
   version: 1.2.0
  git hash: 94ea2ba6
  build no: 0
Application image VALID
Remain in bootloader reason: NoReason
Bootloader:
   version: 1.0.0
  git hash: b3a01818
  build no: 6
Calibration version: 5
Device:
  eco: 0
  rev: 4
  id:  1
SKU: E710
```

4.3.3 Application Image Upload

The application upload example demonstrates uploading an application image to non-volatile memory in the MTI MSx10 SiP, using the built in bootloading capabilities of the device.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the application upload example binary with the image file as an argument:

   ```
   ./build/e710_ref_design/examples/app_upload.bin
   ~/msx10_bundle_v1.02.00/ex10_app_image/ex10_app.bin
   ```

After executing these steps, the binary will run and output the following:

```
Uploading image...
Done
```

```
Application:
   version: 1.2.0
  git hash: 94ea2ba6
  build no: 0
Application image VALID
```

4.3.4 Fixed Q Inventory

The inventory example demonstrates a fixed Q inventory, run successively for a specified time period. For high performance use case, see the next Continuous Inventory section.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   `cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/`

2. Build the SDK:

   `make clean && make -j6`

3. Execute the inventory example binary:

   For low power use case with internal PA of TX RF cable connectivity:

   `./build/e710_ref_design/examples/inventory_fixed_q_internal_pa.bin`

   For high power use case with external PA of TX RF cable connectivity:

   `./build/e710_ref_design/examples/inventory_fixed_q_external_pa.bin`

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting inventory with fixed Q example
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     25235 us] Tx ramp up on channel 909250 kHz
[     25877 us] PowerControl - iterations_taken: 3, final_error: 9,
final_tx_fine_gain: 964
[     27139 us] SJC c: (   17,   -17), a: 2, f: 5, r: (      528,       76),       377
[     27260 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[     34474 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C , RSSI: -4653,
Halted Status: 0
[     35054 us] Inventory round stopped - reason: done, duration_us: 6134,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 1, collided_slots: 0
...
[   9787126 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   9789874 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4592,
Halted Status: 0
[   9793819 us] Inventory round stopped - reason: regulatory, duration_us: 6190,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 1, collided_slots: 0
[   9795318 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   9798346 us] PC: 0x3400 EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4592,
Halted Status: 0
[   9801905 us] Inventory round stopped - reason: done, duration_us: 6083,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 1, collided_slots: 0
[   9811648 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   9815245 us] Inventory round stopped - reason: host, duration_us: 3094,
total_slots: 10, num_slots: 1, empty_slots: 0, single_slots: 0, collided_slots: 0
[   9815558 us] Tx ramp down, reason user
Time elapsed: 10001 ms
Ending inventory example
Total Singulations: 1147
```

4.3.5 Continuous Inventory

The continuous inventory example implements a dynamic Q inventory with a duration expired stop condition. The host can execute this example binary with -h or --help to get the usage information.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the continuous inventory binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/continuous_inventory_internal_pa.bin MODE103
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/continuous_inventory_external_pa.bin MODE103
   ```

After executing these steps, the binary will run and output the following (**NOTE:** the sample output shown below uses a single tag):

```
Starting continuous inventory example
Read rate = 92 - tags: 928 / seconds 10.005 (Mode 103)
```

Tag read data is not printed by default and can be enabled by modifying the ‘verbose’ Boolean in the InventoryHelperParams structure within the example c file.

4.3.6 Halted Read and Write

The halted read and write example demonstrates how to perform Gen2 read and write commands using the halt feature of the MTI Mx10. **NOTE:** This example requires a tag with user memory.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the halted read and write example binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/access_read_write_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

```
./build/e710_ref_design/examples/access_read_write _external_pa.bin
```

After executing these steps, the binary will run and output the following:

```
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     25326 us] Tx ramp up on channel 926250 kHz
[     19467 us] PowerControl - iterations_taken: 4, final_error: 10,
final_tx_fine_gain: 1015
[     27415 us] SJC c: (  -86,     6), a: 2, f: 5, r: (       80,        -4),        57
[     27540 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[     31604 us] PC: 0x3400 EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4606,
Halted Status: 1
[     31641 us] Lmac Halted with handle 21419
Time elapsed: 12 ms
Sending write and read commands
[     38700 us] Gen2Transaction - transaction id: 0, status: ok, num_bits: 33,
data: 0053AB9A
[     39860 us] Gen2Transaction - transaction id: 1, status: ok, num_bits: 49,
data: 0096DE53 AB26
[     39883 us] Lmac Halted with handle 21419
Response 0x96de from Read command matched what was written
Ending write+read sequence example
```

4.3.7 Gen2 Select Command

The select example demonstrates selecting tags using the Gen2v2 select operation for inventory.

To run the example:

1. Navigate to the **msx10_c_dev_kit** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the select command example binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/select_command_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/select_command_external_pa.bin
   ```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting Select command example

Inventory with Sel=ALL, no Select command
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     24494 us] Tx ramp up on channel 926750 kHz
[     25177 us] PowerControl - iterations_taken: 3, final_error: 7,
final_tx_fine_gain: 924
[     26438 us] SJC c: (  -84,    10), a: 2, f: 5, r: (      308,       596),       474
[     26506 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[     32793 us] PC: 0x3400 EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4601,
```

```
Halted Status: 0
[     34266 us] Inventory round stopped - reason: done, duration_us: 6265,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 1, collided_slots: 0
...
[    508400 us] Inventory round stopped - reason: host, duration_us: 4793,
total_slots: 11, num_slots: 1, empty_slots: 0, single_slots: 0, collided_slots: 0
[    508702 us] Tx ramp down, reason user
Time elapsed: 517 ms
Done
The Select mask will be 0xd0c - the CRC of tag with EPC=0xe20030720912015616906627
Inventory with Sel=SL, sending Select with Action001
[    529486 us] Tx ramp up on channel 925250 kHz
[    530336 us] PowerControl - iterations_taken: 4, final_error: 7,
final_tx_fine_gain: 1008
[    531603 us] SJC c: (  -89,    -4), a: 2, f: 5, r: (     -484,      -120),       353
[    531671 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[    533260 us] Gen2Transaction - transaction id: 0, status: no_reply, num_bits: 0,
data:
[    535799 us] PC: 0x3400 EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4639,
Halted Status: 0
[    548780 us] Inventory round stopped - reason: done, duration_us: 6240,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 1, collided_slots: 0
...
[   1014729 us] Gen2Transaction - transaction id: 0, status: no_reply, num_bits: 0,
data:
[   1015670 us] Inventory round stopped - reason: host, duration_us: 624,
total_slots: 1, num_slots: 1, empty_slots: 0, single_slots: 0, collided_slots: 0
[   1015927 us] Tx ramp down, reason user
Time elapsed: 560 ms
Done

Inventory with Sel=~SL, sending Select with Action101
[   1077316 us] Tx ramp up on channel 910750 kHz
[   1078175 us] PowerControl - iterations_taken: 3, final_error: 10,
final_tx_fine_gain: 1047
[   1079433 us] SJC c: (   15,   -32), a: 2, f: 5, r: (     -624,      -376),       515
[   1079501 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xAC, error: 0
[   1081097 us] Gen2Transaction - transaction id: 1, status: no_reply, num_bits: 0,
data:
[   1085078 us] PC: 0x3400 EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4714,
Halted Status: 0
[   1087625 us] Inventory round stopped - reason: done, duration_us: 6213,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 1, collided_slots: 0
...
[   1558924 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   1559896 us] Gen2Transaction - transaction id: 1, status: no_reply, num_bits: 0,
data:
[   1562440 us] PC: 0x3400 EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4667,
Halted Status: 0
[   1563601 us] Inventory round stopped - reason: host, duration_us: 3390,
total_slots: 6, num_slots: 1, empty_slots: 0, single_slots: 0, collided_slots: 0
[   1563870 us] Tx ramp down, reason user
Time elapsed: 520 ms
```

```
Done

Inventory with Sel=SL, sending Select with Action101
[   1586545 us] Tx ramp up on channel 921250 kHz
[   1587475 us] PowerControl - iterations_taken: 5, final_error: 9,
final_tx_fine_gain: 1100
[   1588742 us] SJC c: (  -79,   -44), a: 2, f: 5, r: (       68,      -752),        534
[   1588866 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[   1590464 us] Gen2Transaction - transaction id: 1, status: no_reply, num_bits: 0,
data:
[   1595521 us] Inventory round stopped - reason: done, duration_us: 4742,
total_slots: 16, num_slots: 16, empty_slots: 16, single_slots: 0, collided_slots: 0
...
[   1984743 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   1985713 us] Gen2Transaction - transaction id: 1, status: no_reply, num_bits: 0,
data:
[   1990763 us] Inventory round stopped - reason: host, duration_us: 4734,
total_slots: 16, num_slots: 16, empty_slots: 16, single_slots: 0, collided_slots: 0
[   1991210 us] Tx ramp down, reason regulatory
[   1994949 us] Tx ramp up on channel 909750 kHz
[   1995883 us] PowerControl - iterations_taken: 5, final_error: 6,
final_tx_fine_gain: 1086
[   1997174 us] SJC c: (   17,   -22), a: 2, f: 5, r: (     -172,         4),        122
[   1997246 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[   1998838 us] Gen2Transaction - transaction id: 1, status: no_reply, num_bits: 0,
data:
[   2004364 us] Inventory round stopped - reason: done, duration_us: 5210,
total_slots: 16, num_slots: 16, empty_slots: 15, single_slots: 0, collided_slots: 1
...
[   2068225 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xA8, error: 0x00,
identifier: 0x0000
[   2069197 us] Gen2Transaction - transaction id: 1, status: no_reply, num_bits: 0,
data:
[   2072930 us] Inventory round stopped - reason: host, duration_us: 3418,
total_slots: 11, num_slots: 1, empty_slots: 0, single_slots: 0, collided_slots: 0
[   2073243 us] Tx ramp down, reason user
Time elapsed: 501 ms
Done
Ending Select command example
```

4.3.8 Simple Ramping

The simple ramping example demonstrates ramping the transmitter and allowing the regulatory timers to initiate a ramp down event.

To run the example:

1. Navigate to the **msx10_c_dev_kit** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

```
make clean && make -j6
```

3. Execute the simple ramping example binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/simple_ramping_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/simple_ramping_external_pa.bin
   ```

After executing these steps, the binary will run and output the following:

```
Starting ramp test
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     25143 us] Tx ramp up on channel 920250 kHz
[     25927 us] PowerControl - iterations_taken: 4, final_error: 5,
final_tx_fine_gain: 986
[     27196 us] SJC c: (  -72,   -51), a: 2, f: 5, r: (       40,       576),       408
[     27321 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[    229632 us] Tx ramp down, reason regulatory
Ending ramp test
```

4.3.9 ETSI Burst

The ETSI burst example repeatedly ramps the transmitter up and down while modulating the CW so that the demodulated waveform can be measured.

To run the example:

1. Navigate to the **msx10_c_dev_kit** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the ETSI burst example binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/etsi_burst_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/etsi_burst_external_pa.bin
   ```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting ETSI Burst test
packet type: 1
packet type: 11
packet type: 254
packet type: 5
packet type: 3
packet type: 3
packet type: 3
...
Ending ETSI Burst test
```

4.3.10 Pseudorandom Binary Sequence

The pseudorandom binary sequence (PRBS) example demonstrates the transmission of a PRBS for a specificed time period.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the PRBS example binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/prbs_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/prbs_external_pa.bin
   ```

After executing these steps, the binary will run and output the following:

```
Starting PRBS test
Ending PRBS test
```

4.3.11 Insert EventFifo Packet

The insert EventFifo packet example demonstrates the use of the InsertFifoEvent MTI MSx10 SiP command by inserting custom Event FIFO packets with a test pattern payload.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the insert Event FIFO example binary:

   ```
   ./build/e710_ref_design/examples/insert_event_fifo.bin
   ```

After executing these steps, the binary will run and output the following:

```
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     38231 us] Custom packet - length: 0, data:
[     38425 us] Custom packet - length: 1, data: 12345678
[     38618 us] Custom packet - length: 3, data: 12345678 FEDCBA98 F0001BA1
[     38811 us] Continuous inventory stopped - reason: max duration_us hit, duration_us: 10000000,
number_of_inventory_rounds: 305419896, number_of_tags: 2882400018
```

4.3.12 Continuous Inventory Stop Conditions

The continuous inventory stop conditions example demonstrates multiple dynamic Q inventories, each with a unique stop condition, executed successively.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

```
cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
```

2. Build the SDK:

```
make clean && make -j6
```

3. Execute the continuous inventory binary:

   For low power use case with internal PA of TX RF cable connectivity:

```
./build/e710_ref_design/examples/continuous_inventory_stops_internal_pa.bin
```

   For high power use case with external PA of TX RF cable connectivity:

```
./build/e710_ref_design/examples/continuous_inventory_stops_external_pa.bin
```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting continuous inventory example
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     25397 us] Tx ramp up on channel 926750 kHz
[     26049 us] PowerControl - iterations_taken: 3, final_error: 9, final_tx_fine_gain: 964
[     27311 us] SJC c: (  -84,    11), a: 2, f: 5, r: (     -540,       28),       382
[     27432 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[     30319 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4733,
Halted Status: 0
[     30454 us] Q changed - number of slots: 4, empty slots: 3, single slots: 1, collided slots: 0,
q value: 1, query sent: 0
[     30788 us] Q changed - number of slots: 2, empty slots: 2, single slots: 0, collided slots: 0,
q value: 0, query sent: 0
[     31015 us] Inventory round stopped - reason: done, duration_us: 1932, total_slots: 7, num_slots:
1, empty_slots: 1, single_slots: 0, collided_slots: 0
...
[     62030 us] Inventory round stopped - reason: done, duration_us: 1915, total_slots: 7, num_slots:
1, empty_slots: 1, single_slots: 0, collided_slots: 0
[     62790 us] Continuous inventory stopped - reason: max rounds hit, duration_us: 41774,
number_of_inventory_rounds: 7, number_of_tags: 7
Total Singulations: 7
Time elapsed: 44 ms
[     64437 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00, identifier:
0x0000
[     65717 us] Q changed - number of slots: 3, empty slots: 3, single slots: 0, collided slots: 0,
q value: 1, query sent: 0
[     66045 us] Q changed - number of slots: 2, empty slots: 2, single slots: 0, collided slots: 0,
q value: 0, query sent: 0
[     66270 us] Inventory round stopped - reason: done, duration_us: 1321, total_slots: 6,
num_slots: 1, empty_slots: 1, single_slots: 0, collided_slots: 0
[     67931 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[     69382 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4724,
Halted Status: 0
...
[    243723 us] Inventory round stopped - reason: done, duration_us: 1921, total_slots: 7, num_slots:
1, empty_slots: 1, single_slots: 0, collided_slots: 0
[    244480 us] Continuous inventory stopped - reason: max tags hit, duration_us: 180325,
number_of_inventory_rounds: 42, number_of_tags: 40
Total Singulations: 40
```

```
Time elapsed: 186 ms
[    246126 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[    247409 us] Q changed - number of slots: 3, empty slots: 3, single slots: 0, collided slots: 0,
q value: 1, query sent: 0
[    247742 us] Q changed - number of slots: 2, empty slots: 2, single slots: 0, collided slots: 0,
q value: 0, query sent: 0
[    247968 us] Inventory round stopped - reason: done, duration_us: 1331, total_slots: 6,
num_slots: 1, empty_slots: 1, single_slots: 0, collided_slots: 0
[    249629 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[    251080 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4686,
Halted Status: 0
...
[  10246380 us] Inventory round stopped - reason: regulatory, duration_us: 1728, total_slots: 6,
num_slots: 2, empty_slots: 2, single_slots: 0, collided_slots: 0
[  10246504 us] Tx ramp down, reason regulatory
[  10247145 us] Continuous inventory stopped - reason: max duration_us hit, duration_us: 10001291,
number_of_inventory_rounds: 2301, number_of_tags: 2290
Time elapsed: 10211 ms
Total Singulations: 2290
Ending continuous inventory example
```

4.3.13 Simple Dynamic Q Inventory

The simple dynamic Q inventory example demonstrates a dynamic Q inventory.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   `cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/`

2. Build the SDK:

   `make clean && make -j6`

3. Execute the continuous inventory binary:

   For low power use case with internal PA of TX RF cable connectivity:

   `./build/e710_ref_design/examples/inventory_dynamic_q_internal_pa.bin`

   For high power use case with external PA of TX RF cable connectivity:

   `./build/e710_ref_design/examples/inventory_dynamic_q_external_pa.bin`

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting inventory example
[      4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[     24102 us] Tx ramp up on channel 926250 kHz
[     24778 us] PowerControl - iterations_taken: 3, final_error: 9, final_tx_fine_gain: 931
[     26035 us] SJC c: (  -86,     6), a: 2, f: 5, r: (      340,       216),       285
[     26103 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[     30365 us] Q changed - number of slots: 9, empty slots: 9, single slots: 0, collided slots: 0,
q value: 7, query sent: 0
[     32601 us] Q changed - number of slots: 8, empty slots: 8, single slots: 0, collided slots: 0,
q value: 6, query sent: 0
[     35158 us] Q changed - number of slots: 7, empty slots: 6, single slots: 0, collided slots: 1,
q value: 5, query sent: 0
[     38004 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -4634,
Halted Status: 0
[     38261 us] Q changed - number of slots: 6, empty slots: 5, single slots: 1, collided slots: 0,
q value: 4, query sent: 0
[     39687 us] Q changed - number of slots: 5, empty slots: 5, single slots: 0, collided slots: 0,
q value: 3, query sent: 0
[     40848 us] Q changed - number of slots: 4, empty slots: 4, single slots: 0, collided slots: 0,
q value: 2, query sent: 0
[     41774 us] Q changed - number of slots: 3, empty slots: 2, single slots: 0, collided slots: 1,
q value: 1, query sent: 0
[     42908 us] Q changed - number of slots: 2, empty slots: 1, single slots: 0, collided slots: 1,
q value: 0, query sent: 0
[     43311 us] Inventory round stopped - reason: done, duration_us: 15715, total_slots: 45,
num_slots: 1, empty_slots: 1, single_slots: 0, collided_slots: 0
...
[   9807440 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   9810733 us] Q changed - number of slots: 9, empty slots: 9, single slots: 0, collided slots: 0,
q value: 7, query sent: 0
[   9813017 us] Q changed - number of slots: 8, empty slots: 8, single slots: 0, collided slots: 0,
q value: 6, query sent: 0
[   9813940 us] Inventory round stopped - reason: host, duration_us: 5996, total_slots: 20,
num_slots: 8, empty_slots: 8, single_slots: 0, collided_slots: 0
[   9814288 us] Tx ramp down, reason user
Time elapsed: 10001 ms
Read rate = 53 - tags: 535 / seconds 10.000 (Mode 148)
Ending inventory example
```

4.3.14 Auto-Access Read and Write

The auto-access read and write example demonstrates how to use the auto-access MTI MSx10 SiP feature to perform a Gen2 read and write command sequence. Specifically, random data is written to the user memory page followed by a Gen2 read command to verify what was written to the tag. Auto-access, as opposed to halted access, allows for Gen2 command sequences to be automated by the modem such that the host is not required to manage the modems halted state. **NOTE:** This example requires a tag with user memory.

To run the example:

1. Navigate to the `msx10_c_dev_kit` directory:

   `cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/`

2. Build the SDK:

   `make clean && make -j6`

3. Execute the continuous inventory binary:

   For low power use case with internal PA of TX RF cable connectivity:

   `./build/e710_ref_design/examples/auto_access_read_write_internal_pa.bin`

   For high power use case with external PA of TX RF cable connectivity:

   `./build/e710_ref_design/examples/auto_access_read_write_external_pa.bin`

After executing these steps, the binary will run and output the following:

```
Response 0x63ed from Read command matched what was written
Ending write+read sequence example
```

4.3.15 Measure Chip Sensors

The measure SiP sensors example demonstrates the use of the MTI reader SiPs auxiliary ADC as well as how to access the SJC solution for the existing radio configuration. Measurements are recorded while the radio is transmitting CW and over a frequency and power range specified in the example c file.

To run the example:

1. Navigate to the `msx10_c_dev_kit` directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the continuous inventory binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/measure_chip_sensors_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/measure_chip_sensors_external_pa.bin
   ```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting test characterization of PDETs and SJCs
Carrier frequency (kHz),TX Power Target (cdBm),LO PDET0,LO PDET1,LO PDET2,LO PDET3,RX PDET0,RX
PDET1,RX PDET2,RX PDET3,TestMux0,TestMux1,TestMux2,TestMux3,Temp ADC,LO PDET SUM,RX PDET SUM,SJC
atten,CDAC I value,CDAC I residue,CDAC Q value,CDAC Q residue,
902750,0,119,7,2,12,0,3,4,26,0,0,1,0,304,36,8,0,-24,12,28,28,
902750,300,197,11,1,10,0,4,5,27,0,0,0,0,305,56,8,0,-24,36,28,60,
902750,600,326,21,3,12,0,4,3,26,0,1,0,0,302,95,7,0,-24,-24,27,-52,
902750,900,522,45,3,12,4,4,4,26,0,0,0,0,306,156,8,0,-24,36,28,124,
902750,1200,755,107,3,12,17,5,6,28,0,0,0,1,307,235,14,0,-24,-128,27,-100,
902750,1500,939,240,6,13,86,9,8,28,0,0,0,0,308,334,29,0,-24,-172,27,-108,
902750,1800,988,436,19,15,220,19,6,28,0,0,0,0,310,419,57,0,-25,196,27,-232,
902750,2100,1003,689,48,18,430,45,8,29,0,0,0,0,309,509,106,0,-25,268,27,-308,
902750,2400,1007,927,131,24,717,111,10,30,0,0,0,0,314,615,182,0,-24,-648,27,-144,
902750,2700,62,5,1,10,0,2,6,27,0,0,0,0,304,20,6,0,-25,504,27,-748,
902750,3000,62,5,3,13,0,4,5,26,0,0,2,0,305,21,7,0,-25,672,27,-1152,
903250,0,119,7,1,12,0,4,4,27,0,0,0,0,311,38,8,0,-20,0,26,-24,
903250,300,190,11,1,11,1,5,5,27,0,0,0,0,311,56,8,0,-19,-32,26,-52,
903250,600,319,21,1,12,1,5,5,28,0,0,0,0,311,95,8,0,-19,12,27,68,
...
926750,2400,1001,918,127,27,994,531,44,36,0,0,0,0,335,613,368,0,-83,164,11,692,
926750,2700,63,6,3,14,0,5,6,31,0,0,0,0,327,24,8,0,-84,948,11,-464,
926750,3000,62,5,4,14,0,5,6,29,0,0,0,0,329,22,9,0,-84,1224,11,-628,
927250,0,141,11,4,14,4,7,7,29,0,0,0,0,327,45,9,0,-80,20,15,-12,
927250,300,189,14,3,14,8,8,5,31,0,0,0,0,328,59,8,0,-80,-12,14,-16,
927250,600,311,26,4,14,31,9,7,28,0,0,0,0,328,94,15,0,-80,-36,14,-28,
927250,900,501,51,5,13,111,15,8,29,0,0,0,0,332,154,34,0,-80,20,15,84,
927250,1200,737,111,6,13,259,28,9,30,0,0,0,0,330,230,68,0,-81,116,15,-36,
927250,1500,925,237,11,16,514,67,10,32,0,0,0,0,331,329,130,0,-81,124,15,-68,
927250,1800,981,435,20,18,791,150,13,33,0,0,0,0,332,421,210,0,-81,112,15,-164,
```

```
927250,2100,997,695,54,22,947,298,22,35,0,0,0,0,334,517,289,0,-81,192,15,-180,
927250,2400,1002,922,132,28,992,522,42,37,0,0,0,0,334,616,364,0,-81,192,15,-212,
927250,2700,61,6,3,14,2,5,7,30,0,0,0,0,325,23,8,0,-81,168,15,-292,
927250,3000,63,6,3,13,0,5,8,30,0,0,0,0,333,23,8,0,-81,88,15,-444,
```

4.3.16 Power Modes

The power modes example demonstrates the use of MTI reader SiP power modes by cycling through inventory and one of the power modes for a specified duration. Arguments can be specified when running the binary. See the corresponded examle file to find the accepted arguments.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the continuous inventory binary:

   For low power use case with internal PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/inventory_power_modes_internal_pa.bin
   ```

   For high power use case with external PA of TX RF cable connectivity:

   ```
   ./build/e710_ref_design/examples/inventory_power_modes_external_pa.bin
   ```

After executing these steps, the binary will run and output the following:

```
-T time, in seconds, to run inventory,                  using:   2.0 seconds
-t time, in seconds, to paused in low power mode,       using:   2.0 seconds
-n the number of inventory -> lower power iterations,   using:   2   cycles
-p mode, the low power mode to use,                     using:   1   PowerModeOff
---------- iteration:  1 /  2:
inventory power mode: 4, PowerModeReady
continuous inventory, duration:     2.000
Tag Read rate:          82
Number of tags read:    164
Numbers of seconds:       2.010
RF Mode:                 11
low power mode: 1, PowerModeOff
---------- iteration:  2 /  2:
inventory power mode: 4, PowerModeReady
continuous inventory, duration:     2.000
Tag Read rate:          84
Number of tags read:    168
Numbers of seconds:       2.002
RF Mode:                 11
low power mode: 1, PowerModeOff
```

4.3.17 SPI Transfer Test

The SPI transfer test example demonstrates the use of the TransferTest command to test the physical SPI interface that connects the host to the MTI reader SiP. See the corresponded examle file to find the accepted arguments.

To run the example:

1. Navigate to the `msx10_c_dev_kit` directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the SPI transfer test example binary with 20 iterations:

   ```
   ./build/e710_ref_design/examples/test_spi_transfer.bin 20
   ```

After executing these steps, the binary will run and output the following:

```
Test transfer 0 (size 0)
Test transfer 2 (size 1)
Test transfer 4 (size 2)
Test transfer 6 (size 3)
Test transfer 8 (size 4)
Test transfer 10 (size 5)
Test transfer 12 (size 6)
Test transfer 14 (size 7)
Test transfer 16 (size 8)
Test transfer 18 (size 9)
Pass
```

4.4 Calibration Information

The MTI MSx10 SiP has a dedicated flash page for calibration and characterization data. It is flash page **`0x3`**, and it has 2048 bytes of capacity. The arrangement of data in this flash page is arbitrary, and can be configured per-application to optimize reader calibration performance. MTI provides an example format of the calibration data, defined in the **`cal_info_page_v5.json`** JSON file. The host code must know how to interpret the calibration data format, so it is important to properly record and manage the format, especially if multiple versions are developed.

The calibration flash page is written in using the **`write_cal_info_page.bin`**. Calibration is read out of flash using the **`read_cal_info_page.bin`**. The calibration flash page is erased using the **`erase_cal_info_page.bin`**.

There are two factory **`<board id number>.json`** files, one is calibration for low power with internal PA, and another is calibration for high power with external PA, stored in the **`~/ex10_board_cal/`** directory.
There are two **`H5001.json`** files with default power, **`15001.json`** and **`25001.json`**, stored in the same directory for general use before power calibration.
The filename format of factory JSON file is described below:

HVRSS.json
- H: Hardware configuration, 1 = low power with internal PA, 2 = high power with external PA
- V: Version number, 5 = V5 for v1.0.12 SDK and later (supports RSSI calibration)
- R: Reserved for future use, here are ‘0’
- SS: Serial number, here are the last two codes of serial number

The calibration data of low power with internal PA has been saved in flash page as default.

4.4.1 Print Calibration Information

The print calibration example prints the calibration parameters stored in the MTI MSx10 SiP for the respective calibration scheme used. The calibration version stored on the MTI MSx10 SiP, which can be checked with the get version information example in Get Version Information section. The value of user_board_id field in the calibration information is equal to the file name of HVRSS.json.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:
   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```
2. Build the SDK:
   ```
   make clean && make -j6
   ```
3. Execute the print calibration example binary:
   ```
   ./build/e710_ref_design/examples/print_calibration_v5.bin
   ```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Calibration:
   Version: file: 5, pwr_det: 5, fwd_pwr: 1, pwr_det_temp: 2
          fwd_pwr_temp: 1, pwr_det_freq: 2, fwd_pwr_freq: 1
   user_board_id: 15001
   tx_scalar_cal: 1152
   RFFilter:
         Lower Band: [865.700, 867.500]
         Upper Band: [902.750, 927.250]
   ValidPdetAdcs:
         Valid min ADC: 150
```

```
        Valid max ADC: 915
ControlLoopParams:
        Loop Gain Divisor: 800
        Error Threshold: 10
        Max Iterations: 10
PdetAdcLut:
       Lower Band                    Upper Band
      pdet0     pdet1     pdet2       pdet0     pdet1     pdet2
[ 0]:   978       976       379         977       980       432
[ 1]:   977       968       297         975       973       341
...
[30]:     7        56        20           7        56        20
FwdPowerCoarsePwrCal:
        Lower Band    Upper Band
[ 0]:     27.980        28.140
[ 1]:     26.661        26.775
...
[30]:     -7.553        -7.779
FwdPowerTempSlope:
        Lower Band    Upper Band
 slope:   -0.039        -0.039
CalTemp ADC:
        Lower Band    Upper Band
            313           315
LoPdetTempSlope:
        Lower Band
 slope:    0.020,    0.013,    0.012
        Upper Band
 slope:    0.020,    0.013,    0.012
LoPdetFreqLut:
    Lower Band                        Upper Band
     shifts0   shifts1   shifts2      shifts0   shifts1   shifts2
[ 0]:    -5      -36        0          -19       -14         0
[ 1]:     0      -37        0           -8        -4         0
[ 2]:    -5      -17        0           -1         3         0
[ 3]:     6        2        0            3         4         0
LoPdetFreqs:
        Lower Band    Upper Band
         865.700       902.750
         866.300       910.250
         866.900       920.250
         867.500       927.250
FwdPwrFreqLut:
        Lower Band    Upper Band
          -0.307        -0.611
          -0.050        -0.668
           0.398        -0.546
           0.834        -0.773
DcOffsetCal:
[ 0]:    6888
[ 1]:    6888
...
[30]:   23170
...
RssiTempSlope:       RssiTempIntercept:
         0.000                   299
```

4.4.2 Read Calibration Info Page

The read calibration info page example download the calibration parameters stored in the MTI MSx10 SiP.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the read calibration information page example binary:

   ```
   ./build/e710_ref_design/examples/read_cal_info_page.bin
   ```

After executing these steps, the binary will attempt to download calibration informantion page data and save it to the calibration JSON file, and will report on the results:

```
MSx10 development kit calibration data reading out example


Enter calibrated board ID number (max 65535): 15001
Using calibration version 5
Read out power data from the calibration info page
Write into the /home/pi/ex10_board_cal/15001.json file
```

4.4.3 Write Calibration Info Page

The write calibration info page example upload the calibration parameters to MTI MSx10 SiP.

To run the example:

1. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the write calibration information page example binary:

   ```
   ./build/e710_ref_design/examples/write_cal_info_page.bin
   ```

After executing these steps, the binary will attempt to upload the calibration JSON file, and will report on the results:

```
MSx10 development kit calibration data writing into example


Enter calibrated board ID number (max 65535): 15001
Read the /home/pi/ex10_board_cal/15001.json file
Write power data into the calibration info page
```

4.4.4 Erase Calibration Info Page

The erase calibration info page example erase the calibration parameters stored in the MTI MSx10 SiP.

To run the example:

1. Navigate to the `msx10_c_dev_kit` directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

2. Build the SDK:

   ```
   make clean && make -j6
   ```

3. Execute the read calibration information page example binary:

   ```
   ./build/e710_ref_design/examples/erase_cal_info_page.bin
   ```

After executing these steps, the binary will attempt to erase calibration informantion page data and will report on the results:

```
MSx10 development kit calibration data erasing example

Erase calibration info page done
```

4.5 Listen Before Talk (LBT)

There is a new operation in firmware (ListenBeforeTalkOp (0xBA)) that checks the power level around the currently locked frequency, and reports it back to the host. An example demonstrates how to use this Op, but it is a modified version of the ex10_helpers layer, ex10_lbt_helpers.c and ex10_lbt_helpers.h. In a future release, MTI will likely integrate the example and helpers.

4.5.1 LBT with a large interferer present

Here is an execution of the example, with an argument specifying the carrier frequency in kHz, and a large interferer present which causes the example to abort inventory immediately.

To run the example:

4. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

5. Build the SDK:

   ```
   make clean && make -j6
   ```

6. Execute the print calibration example binary:

   ```
   ./build/e710_ref_design/examples/lbt_inventory_internal_pa.bin 920400
   ```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting inventory with LBT example
[     4048 us] Hello from E710, Reset reason: 0x00, cond: 1
LBT failed to pass at 920400 kHz due to a noisy environment
Detected power level at antenna port (cdBm): -5537
Threshold (cdBm): -7400
Read rate = 0 - tags: 0 / seconds 10.000 (Mode 148)
Ending inventory example
No tags found in inventory
```

4.5.2 LBT without a large interferer present

Here is an execution of the example, without a large interferer present.

To run the example:

7. Navigate to the **`msx10_c_dev_kit`** directory:

   ```
   cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
   ```

8. Build the SDK:

   ```
   make clean && make -j6
   ```

9. Execute the print calibration example binary:

   ```
   ./build/e710_ref_design/examples/lbt_inventory_internal_pa.bin
   ```

After executing these steps, the binary will run and output the following (excerpt shown):

```
Starting inventory with LBT example
[     4048 us] Hello from E710, Reset reason: 0x00, cond: 1
[    47101 us] Tx ramp up on channel 918000 kHz
[    47809 us] PowerControl - iterations_taken: 3, final_error: 9, final_tx_fine_gain: 949
[    49072 us] SJC c: (  -47,   -48), a: 2, f: 5, r: (      704,      -472),       599
[    49196 us] Aggregate op - op_run_count: 8, write_count: 16, insert_fifo_count: 0,
```

```
final_buffer_byte_index: 164, total_jump_count: 0, last_inner_op: run: 0xac, error: 0x00,
identifier: 0x0000
[     53421 us] Q changed - number of slots: 9, empty slots: 9, single slots: 0, collided slots: 0,
q value: 7, query sent: 0
[     55681 us] Q changed - number of slots: 8, empty slots: 8, single slots: 0, collided slots: 0,
q value: 6, query sent: 0
[     57713 us] Q changed - number of slots: 7, empty slots: 7, single slots: 0, collided slots: 0,
q value: 5, query sent: 0
[     59455 us] Q changed - number of slots: 6, empty slots: 6, single slots: 0, collided slots: 0,
q value: 4, query sent: 0
[     60907 us] Q changed - number of slots: 5, empty slots: 5, single slots: 0, collided slots: 0,
q value: 3, query sent: 0
[     62088 us] Q changed - number of slots: 4, empty slots: 4, single slots: 0, collided slots: 0,
q value: 2, query sent: 0
[     62942 us] Q changed - number of slots: 3, empty slots: 3, single slots: 0, collided slots: 0,
q value: 1, query sent: 0
[     63603 us] Q changed - number of slots: 2, empty slots: 2, single slots: 0, collided slots: 0,
q value: 0, query sent: 0
[     63975 us] Inventory round stopped - reason: done, duration_us: 13456, total_slots: 45,
num_slots: 1, empty_slots: 1, single_slots: 0, collided_slots: 0
[     66176 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[     69601 us] Q changed - number of slots: 9, empty slots: 9, single slots: 0, collided slots: 0,
q value: 7, query sent: 0
[     71948 us] Q changed - number of slots: 8, empty slots: 8, single slots: 0, collided slots: 0,
q value: 6, query sent: 0
[     74445 us] Q changed - number of slots: 7, empty slots: 6, single slots: 0, collided slots: 1,
q value: 5, query sent: 0
[     76235 us] Q changed - number of slots: 6, empty slots: 6, single slots: 0, collided slots: 0,
q value: 4, query sent: 0
[     78479 us] PC: 0x3400, EPC: 0x E2003072 09120156 16906627, CRC: 0x0D0C, RSSI: -3267,
Halted Status: 0
[     79049 us] Q changed - number of slots: 5, empty slots: 4, single slots: 1, collided slots: 0,
q value: 3, query sent: 0
[     80190 us] Q changed - number of slots: 4, empty slots: 4, single slots: 0, collided slots: 0,
q value: 2, query sent: 0
[     81077 us] Q changed - number of slots: 3, empty slots: 3, single slots: 0, collided slots: 0,
q value: 1, query sent: 0
[     81724 us] Q changed - number of slots: 2, empty slots: 2, single slots: 0, collided slots: 0,
q value: 0, query sent: 0
[     82093 us] Inventory round stopped - reason: done, duration_us: 15352, total_slots: 45,
num_slots: 1, empty_slots: 1, single_slots: 0, collided_slots: 0
...
[   9814125 us] Aggregate op - op_run_count: 1, write_count: 1, insert_fifo_count: 0,
final_buffer_byte_index: 9, total_jump_count: 0, last_inner_op: run: 0xa8, error: 0x00,
identifier: 0x0000
[   9817413 us] Q changed - number of slots: 9, empty slots: 9, single slots: 0, collided slots: 0,
q value: 7, query sent: 0
[   9818063 us] Inventory round stopped - reason: host, duration_us: 3430, total_slots: 11,
num_slots: 9, empty_slots: 9, single_slots: 0, collided_slots: 0
[   9818319 us] Tx ramp down, reason user
Time elapsed: 10001 ms
Read rate = 7 - tags: 70 / seconds 10.000 (Mode 148)
Ending inventory example
```

5 RASPBERRY PI 4 CONFIGURATION

5.1 Raspberry Pi 4 Out of Box Configuration

The MTI MSx10 development kit has been tested with a Raspberry Pi host running Raspberry Pi OS and Linux kernel v5.10.17. The modifications that have been made to the preconfigured boards are detailed below.

1. The Linux kernel v5.10.17 was used to test this configuration. Install v5.10.17 with the following procedures:
   a. Operate the Raspberry Pi Imager application to erase a micro SD card, and then burn the Raspberry Pi OS with Linux kernel v5.10.17.
   b. Place an empty file named 'ssh' in the root directory of the newly installed Raspberry Pi OS on new micro SD card and it will enable SSH.
   c. Activate the DHCP service, e.g. router device, Tftpd64 application, it should assign the Raspberry Pi an IP address or the special **`192.168.0.10`**.
   d. Login the Raspberry Pi with default username **`pi`** and password **`raspberry`** through SSH.
2. Configure the static IP address with the following modifications:
   a. Edit the dhcpcd.conf file with the following command.

      ```
      sudo nano /etc/dhcpcd.conf
      ```

   b. Add the settings at the end of file with following description.

      ```
      # Static IP configuration
      interface eth0
      static ip_address=192.168.0.10/24
      static routers=192.168.0.1
      static domain_name_servers=192.168.0.1 8.8.8.8
      ```

The Raspberry Pi's SPI interface must be enabled for the MTI reader SiP host library to use. The SPI interface and required settings can be configured within the raspi-config application.

3. Start the raspi-config application and configure with the following procedures:
   a. This application can be started from a terminal window on the Raspberry Pi with the following command.

      ```
      sudo raspi-config
      ```

   b. At the main menu, select "**`1 System Options`**"
      i. Select "**`S3 Password`**"
         New password: "**`rpi2020`**" → Retype new password: "**`rpi2020`**"
      ii. Select "**`S1 Wireless LAN`**"
         Country: "**`TW`**" → SSID: "**`MTISoftware_2G`**" → passphrase: "**`xxxxxxxxxxx`**"
   c. At the main menu, select "**`3 Interfacing Options`**"
      i. Select "**`P2 SSH`**" → Select "**`Yes`**"
      ii. Select "**`P3 VNC`**" → Select "**`Yes`**"
      iii. Select "**`P4 SPI`**" → Select "**`Yes`**"
      iv. Select "**`P6 Serial Port`**" → Select "**`No`**" → Select "**`Yes`**"
         (**Note:** disable the serial login shell and enable the serial interface)

d. At the main menu, select “**`5 Localisation Options`**”

   i. Select “**`L1 Locale`**”
      Choose “**`en_US.UTF-8`**” and “**`zh_TW.UTF-8`**” → Default “**`en_US.UTF-8`**”

   ii. Select “**`L2 Timezone`**”
       Select “**`Asia`**” → Select “**`Taipei`**”

   iii. Select “**`L4 WLAN Country`**”
        Select “**`TW`**”

e. At the main menu, select “**`6 Advanced Options`**”

   i. Select “**`A1 Expand Filesystem`**”

f. At the main menu, select “**`Finish`**” button

   i. Select “**`Yes`**” to reboot.

4. The Raspberry Pi’s system date and time must be corrected before installing Linux packages with the following commnad.

```
sudo date -s "yyyy-mm-dd hh:mm:ss"
```

5. Various Linux packages are required and can be installed from a terminal window on the Raspberry Pi (internet connection required):

```
sudo apt-get install libatlas-base-dev python3-libgpiod libgpiod-dev gpiod
```

6. Modify the device tree overlay.

   a. Edit the config.txt file wit the following command.

   ```
   sudo nano /boot/config.txt
   ```

   b. Add the appropriate setting in the [all] section of file with following description.

   ```
   [all]
   #dtoverlay=vc4-fkms-v3d
   dtoverlay=spi0-2cs,cs1_pin=27
   enable_uart=1
   ```

7. A restart of the Raspberry Pi is required after making these configuration changes.

```
sudo reboot
```

Install the MTI reader SiP SDK using the instructions in the Updating the MTI Reader SiP SDK section.

5.2 Updating the MTI Reader SiP SDK

MTI reader SiP SDK updates are installed as described in the steps below.

1. Connect to your Raspberry Pi using one of the methods below:
   a. Remotely via SSH using a tool like MobaXterm
   b. Locally using a keyboard, mouse, and monitor
2. Open a command line if your connection method hasn’t given you one automatically
3. Navigate to the home directory:

   ```
   cd ~
   ```

4. Get the version 1.02.00 of the MTI reader SiP SDK release package ‘**`msx10_bundle_v1.02.00_datecode.zip`**’ to your computer or to the Pi

5. Extract the MTI reader SiP SDK to the **`~/msx10_bundle_v1.02.00/`** directory on the Raspberry Pi
    a. Locally in the console:
    ```
    unzip msx10_bundle_v1.02.00_datecode.zip -d ~/
    ```
    b. Alternatively: Extract on your remote PC and copy over to the **`~/msx10_bundle_v1.02.00/`** directory on the Raspberry Pi via USB drive or FTP server (MobaXterm is a great tool for this)
6. Navigate to the **`msx10_c_dev_kit`** directory and build the SDK.
    a. Navigate to the **`msx10_c_dev_kit`** directory:
    ```
    cd ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/
    ```
    b. Build the SDK:
    ```
    make clean && make -j6
    ```
7. Update the firmware version on the MTI MSx10 SiP using the application upload example:
    ```
    cd
    ~/msx10_bundle_v1.02.00/msx10_c_dev_kit/build/e710_ref_design/examples/

    ./app_upload.bin ~/msx10_bundle_v1.02.00/ex10_app_image/ex10_app.bin
    ```

5.3 Building a Python Virtual Environment

It is highly recommended that a Python virtual environment is created before executing the calibration flash page access scripts. Working within a virtual environment helps to maintain a consistent environment of version specified Python modules (see **`py3_requirements-lock.txt`** for modules and versions). Creating the Python virtual environment only needs to be done once and activated at the instantiation of each terminal session. Creating the Python virtual environment can be done with the following commands (internet connection required):

```
cd ~/ex10_bundle_v0.08.00/ex10_dev_kit/

source source_me_for_rpi_py3_venv.sh
```

Creating the virtual environment will also activate the virtual environment. Now that it has been created, it only needs to be activated when a new terminal session is instantiated. An existing Python virtual environment can be activated with the following commands:

```
cd ~/ex10_bundle_v0.08.00/ex10_dev_kit/

source py3_venv/bin/activate
```

The terminal prompt will be prepended with the virtual environment name (in parenthesis) when the virtual environment is active:

```
(py3_venv) pi@raspberrypi:~/ex10_bundle_v0.08.00/ex10_dev_kit $
```

To deactivate the Python virtual environment, execute the following command within an active virtual environment:

```
deactivate
```

6 REFERENCE DOCUMENTS

Related documents are listed below.

Table 3: Reference Documents

Document	Description
MTI MSx10 Development Kit User's Guide (this document)	Documents how to use the complete development kit, SDK, and host examples, including Quick Start Guide.
MTI MSx10 Development Board Application Note	Documents the MTI MSx10 development board hardware, circuit topologies, design performance, and potential modifications.
MTI MSx10 RFID Reader SiP Datasheet	Documents the MTI MSx10 SiP, including electrical and mechanical specifications.
MTI MSx10 RFID Reader Calibration Application Note	Documents an example procedure and background to calibrate an MTI MSx10 SiP based RAIN RFID reader.
MTI MSx10 MCU Host Porting Application Note	Demonstrates a RAIN RFID inventory example running on a NUCLEO-L433RC-P microcontroller development board host, using the C library included with the MTI reader SiP SDK.

7 DOCUMENT CHANGE LOG

Table 4: Document Change Log

Version	Date	Description
0.6	2021-01-26	● First release of preliminary version.
0.8	2021-07-21	● Updated for version 0.8.0 Beta SDK and firmware. ● Changed references of 'Reference Design' to 'Development Board'. ● Replace python SDK content with C code. ● Removed python SDK content, but keep python code of read/write calibration information page. ● Added details on C static libraries produced during compilation. ● Added description and pictures to MTI reader SiP SDK section. ● Added hardware/software compatibility matrix.
1.0	2021-10-13	● Updated for version 1.00.12 SDK and firmware. ● Calibration v5 added (supports RSSI calibration). ● Support added for newer Raspberry Pi Model 4 version (RPi BCM2711 GPIO label added).
1.1	2022-01-20	● Updated for version 1.01.00 SDK and firmware. ● Add utility example usage information in C Host Examples section. ● Add two documents in REFERENCE DOCUMENTS section, ○ MTI MSx10 RFID Reader Calibration Application Note. ○ MTI MSx10 MCU Host Porting Application Note.
1.1b	2022-03-23	● Support Raspberry Pi 4 Model B Rev 1.2 and 1.4 both hardware revisions. ● Add description of two default power JSON files. ● Added details on on Raspberry Pi out of box configuration. ● Add an update code to version control information.
1.2a	2022-08-19	● Updated for version 1.02.00 SDK and firmware. ● Removed calibration v4 and related description. ● Added C examples description. ○ Added Continuous Inventory Stop Conditions section. ○ Added Simple Dynamic Q Inventory section. ○ Added Auto-Access Read and Write section. ○ Added Measure Chip Sensors section. ○ Added Power Modes section. ○ Added SPI Transfer Test section. ● Added Listen Before Talk (LBT) section. ● Added read/ write/ erase calibration information C examples description. ● Renamed C examples section for clarity. ○ Renamed Inventory to Fixed Q Inventory. ○ Renamed Gen2 Read and Write Access to Halted Read and Write. ○ Renamed Insert Event FIFO Packet to Insert EventFifo Packet. ○ Renamed Select Command to Gen2 Select Command.

8 NOTICES

MTI gives no representation or warranty, express or implied, for accuracy or reliability of information in this document. MTI reserves the right to change its products and services and this information at any time without notice.