APPLICATION NOTE:
SX-ULPGN-BTZ
Memory Consumption

Silex Technology America
201 East Sandpointe, Suite 245
Santa Ana, CA 92707
Revision 1.0, May 28, 2019

Revision History

Rev. No.	Date	Revision by	Description
1.0	May 28, 2019	T.Nakase	Initial draft

Table of Contents

1. Scope

This application note provides a methodology for measuring memory consumption data when executing SX-ULPGN-BTZ-EVK QCLI_demo projects.

2. References

2.1 SX-ULPGN-BTZ QSG

2.1.1 SX-ULPGN-BTZ Development Quick Start Guide, 140-00217-100 v1.4

2.2 QDN Programmer's Guide

2.2.1 QCA402x (CDB2x) Programmer's Guide, 80-YA121-142 Rev. D

2.3 QDN QAPI Specification

2.3.1 QCA402x QAPI Specification, 80-Y9381-7 Rev. F

3. Equipment

3.1 Hardware

- The EVK board, SX-ULPGN-BTZ EVK (WCBN3516A_EVB V01) x2
- IEEE 802.11 Access Point (Linksys EA6350)
- Shield Box (Ramsey Electronics, LLC STE4400)
- Host PC
- USB 2.0 Cable (Type A male – Type B male) x4
- USB Gender Changer (Type B female – Type A male) x4
- USB Hub
- Jumper Cap x18
- Jumper Cable (female - female) x6

3.2 Host PC Configuration

- Intel Core i7-4790 Processor @ 3.60 GHz
- 8 GB RAM
- 160GB HDD
- USB 2.0/3.0 x3
- Gigabyte Ethernet Port x1
- Windows 10 Professional
- Username: silex, Account type: Administrator

4. Software Installation

4.1 SX-ULPGN-BTZ Development Environment

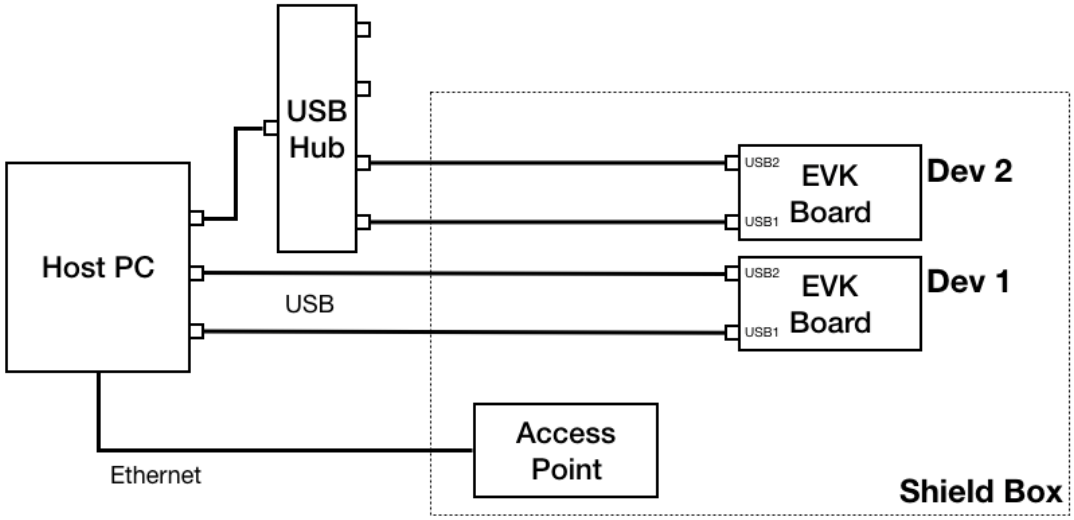
Follow Section 4 of the **SX-ULPGN-BTZ QSG** document to setup a development environment to your **Host PC**.

4.2 Program Firmware Image

You need x2 **EVK boards**. Follow Sections 5 - 8 in the **SX-ULPGN-BTZ QSG** to build and flash the QCLI_demo.

5. Measurement System Setup

Place the **EVK boards** and **Access Point** in the **Shield Box** as shown in the diagram below.



6. Memory Allocation

6.1 System Overview

SX-ULPGN-BTZ SoC has three heterogeneous CPU cores: ARM Cortex-M4F for application, ARM Cortex-M0 for the IEEE802.15.4/BLE stack, and Xtensa for the WiFi stack. All three of these heterogeneous CPU cores enable programmers to allocate memory from the ARM Cortex-M4F RAM and SoC on-chip quad SPI flash ROM to their application.

To conserve power consumption, programmers can put SX-ULPGN-BTZ into low power states from the fullest power mode to the lowest power consumption mode. These modes include Full Operation Mode (FOM), Sensor Operation Mode (SOM), and Minimum Operation Mode (MOM), and can cut down the power supply through the use of pre-selected RAM regions and unused components.

These operation modes have dedicated RAM regions, respectively. Programmers can only allocate application memory for FOM and SOM. MOM is merely always-on (AON) RAM that saves application data during low power mode. The allocation of the application code is not possible in this mode.

Due to the design decisions mentioned above, programmers should be aware of memory consumption in FOM/SOM RAM regions and XIP on-chip quad SPI flash ROM regions.

6.2 Memory Mappings

6.2.1 Code Segment

QCA4020 SDK build script generate GCC linker script suitable for your application and RTOS with python script:

`<SDK_source>\target\build\script\linkerScripts\MakeLinkerScript.py`

Python script has a very straightforward operation, but the ultimate source of memory allocation comes down to following **SECTIONS** and **MEMORY** commands using the linker script template:

`<SDK_source>\target\bin\cortex-m4\threadx\DefaultTemplateLinkerScript.ld`

SECTIONS	MEMORY	Default section assignment
RAM_FOM_APPS_RO_REGION	RAM_FOM_APPS_RO_MEMORY	*(.conststring) *(.constdata) *(.rodata*)
RAM_FOM_APPS_RW_REGION	RAM_FOM_APPS_DATA_MEMORY	*(.data)
RAM_FOM_APPS_ZI_REGION	RAM_FOM_APPS_DATA_MEMORY	*(.bss) *(COMMON)
RAM_SOM_APPS_RO_REGION	RAM_FOM_APPS_RO_MEMORY	-
RAM_SOM_APPS_RW_REGION	RAM_SOM_APPS_DATA_MEMORY	-
RAM_SOM_APPS_ZI_REGION	RAM_SOM_APPS_DATA_MEMORY	-
XIP_OEM_RO_REGION	XIP_RO_MEMORY	*(.text)

Programmers can override the default section assignment using the per-project configuration file:

`<SDK_source>\target\quartz\demo\QCLI_demo\build\gcc\app.config`

The entry consists of program section names (library, object file, program section name) and flags:

[Library]:[Object]([Section]) [Address Space] [Operation Mode] [Origin] [Protocol] [Special]

- Section Name
  - **[Library]** Library File
  - **[Object]** Object File
  - **[Section]** Program Section
- Flags
  - **[Address Space] RAM** or **XIP**
  - **[Operation Mode] FOM** or **SOM**
  - **[Origin] SYS** or **APPS** (synonym: **APP**)

- **[Protocol] RO**, **RW** or **ZI**
- **[Special]** Special section selector (see linker script template for all the selections available)

Programmers can specify the above flags in any order. If the flag is specified twice, the last one seen will take effect. If the flag is not fully specified, python script determines the rest based on the flag supplied and ELF header of the object file.

- **[Protocol]** is determined by the ELF header of the specified program section
- If **[Address Spec]** is missing, set it to **XIP** if **[Protocol]** is **RO**, set it to **RAM** otherwise
- If **[Operation Mode]** is missing, set it to **FOM**
- If **[Origin]** is missing, set it to **APP**
- If **[Special]** is missing, set it to **NORMAL**

Giving **[Address Space]**, **[Operation Mode]** and **[Origin]** flags should suffice in most cases when the programmer needs to specify their program sections in the **app.config** file. This script also applies to the following reconcile rule.

- If **[Protocol]** is **RW** or **ZI**, set **[Address Space]** to **RAM**

This last rule is particularly useful when programmers are assigning all but the .data and .bss program sections of the object file to **XIP**. The programmer just has to put the whole object file into **XIP** with the app.config file. This rule will automatically divert writable sections to **RAM**.

All in all, the sections programmers have specified in the **app.config** file should be assigned to the following sections.

SECTIONS	MEMORY	app.config Flag
RAM_FOM_APPS_RO_REGION	RAM_FOM_APPS_RO_MEMORY	RAM FOM APPS RO NORMAL
RAM_FOM_APPS_RW_REGION	RAM_FOM_APPS_DATA_MEMORY	RAM FOM APPS RW NORMAL
RAM_FOM_APPS_ZI_REGION	RAM_FOM_APPS_DATA_MEMORY	RAM FOM APPS ZI NORMAL
RAM_SOM_APPS_RO_REGION	RAM_SOM_APPS_RO_MEMORY	RAM SOM APPS RO NORMAL
RAM_SOM_APPS_RW_REGION	RAM_SOM_APPS_DATA_MEMORY	RAM SOM APPS RW NORMAL
RAM_SOM_APPS_ZI_REGION	RAM_SOM_APPS_DATA_MEMORY	RAM SOM APPS ZI NORMAL
XIP_OEM_RO_REGION	XIP_RO_MEMORY	XIP FOM APPS RO NORMAL

6.2.2 Stack and Heap

FOM and SOM have dedicated sections for their own stack and heap. These allocations can be read in the linker script template.

Segment	SECTIONS	MEMORY
FOM Heap	None	Start at the end of RAM_FOM_APPS_DATA_MEMORY to RAM_FOM_BSP_STACK_MEMORY
FOM Stack	RAM_FOM_BSP_STACK_ZI_REGION	RAM_FOM_BSP_STACK_MEMORY
SOM Heap	RAM_SOM_BSP_HEAP_ZI_REGION	RAM_SOM_BSP_HEAP_MEMORY
SOM Stack	RAM_SOM_BSP_STACK_ZI_REGION	RAM_SOM_BSP_STACK_MEMORY

7. QCLI_demo Measurement

7.1 Code Segment

Read the total size of the code segments allocated to each section using the following command:

```
arm-none-eabi-objdump -h
<SDK_source>\quartz\demo\QCLI_demo\build\gcc\output\Quartz.elf
```

MEMORY	Size [KiB]	SECTION	Used {KiB]
RAM_FOM_APPS_RO_MEMORY	232	RAM_FOM_APPS_RO_REGION	189
RAM_FOM_APPS_DATA_MEMORY	64	RAM_FOM_APPS_RW_REGION	3.5
		RAM_FOM_APPS_ZI_REGION	19
RAM_SOM_APPS_RO_MEMORY	22	RAM_SOM_APPS_RO_REGION	3
RAM_SOM_APPS_DATA_MEMORY	5	RAM_SOM_APPS_RW_REGION	0.1
		RAM_SOM_APPS_ZI_REGION	0.2
XIP_RO_MEMORY	3072	XIP_OEM_RO_REGION	382

7.2 Heap

Retrieve the used and free amount of **FOM Heap** with **qapi_Heap_Status()**. QAPI does not have a similar API for SOM, and no API is available for the stack.

Memory	Size [KiB]	Note
Start at the end of RAM_FOM_APPS_DATA_MEMORY	126 + 41 (= rest of RAM_FOM_APPS_DATA_MEMORY)	FOM Heap

to RAM_FOM_BSP_STACK_MEMORY		
RAM_FOM_BSP_STACK_MEMORY	2	FOM Stack
RAM_SOM_BSP_HEAP_MEMORY	4	SOM Heap
RAM_SOM_BSP_STACK_MEMORY	2	SOM Stack

7.2.1 TCP RX

SX-ULPGN-BTZ EVK (Dev 1) **IP address = 192.168.77.1**	**Host PC** **IP address = 192.168.77.10**	**Heap Used (Dev1) [byte]**
After bootup	n/a	77624
> wlan WLAN> enable WLAN> setdevice 1	n/a	99440
WLAN> connect <AP SSID>	n/a	99440
WLAN> up > net NET> ifconfig 192.168.77.1 255.255.255.0 NET> ping 192.168.77.10	n/a	99440
NET> benchrx tcp 9000	n/a	103680
n/a	$ ath_console.exe tx 192.168.77.1 9000 tcp 1000 0 150 100 v4	105632
n/a	Stop ath_console.exe	103680
NET> benchquit	n/a	99440
NET> up > wlan WLAN> disconnect	n/a	99440
WLAN> disable	n/a	83016

7.2.2 TCP TX

SX-ULPGN-BTZ EVK (Dev 1) IP address = 192.168.77.1	Host PC IP address = 192.168.77.10	Heap Used (Dev 1) [byte]
After bootup	n/a	77624
> wlan WLAN> enable WLAN> setdevice 1	n/a	99440
WLAN> connect <AP SSID>	n/a	99440
WLAN> up > net NET> ifconfig 192.168.77.1 255.255.255.0 NET> ping 192.168.77.10	n/a	99440
n/a	$ ath_console.exe rx 192.168.77.10 6000 tcp v4	99440
NET> benchtx 192.168.77.10 6000 tcp 1000 0 150 100	n/a	104256
NET> benchquit	n/a	99440
NET> up > wlan WLAN> disconnect	n/a	99440
WLAN> disable	n/a	83016

7.2.3 UDP RX

SX-ULPGN-BTZ EVK (Dev 1) IP address = 192.168.77.1	Host PC IP address = 192.168.77.10	Heap Used (Dev 1) [byte]
After bootup	n/a	77624
> wlan WLAN> enable WLAN> setdevice 1	n/a	99440
WLAN> connect <AP SSID>	n/a	99440
WLAN> up > net NET> ifconfig 192.168.77.1 255.255.255.0 NET> ping 192.168.77.10	n/a	99440
NET> benchrx udp 9000	n/a	105032
n/a	$ ath_console.exe tx 192.168.77.1 9000 udp 1000 0 150 100 v4	105032
n/a	Stop ath_console.exe	105032
NET> benchquit	n/a	99440
NET> up > wlan WLAN> disconnect	n/a	99440
WLAN> disable	n/a	83008

7.2.4 7UDP TX

SX-ULPGN-BTZ EVK (Dev 1) IP address = 192.168.77.1	Host PC IP address = 192.168.77.10	Heap Used (Dev 1) [byte]
After bootup	n/a	77608
> wlan WLAN> enable WLAN> setdevice 1	n/a	99424
WLAN> connect <AP SSID>	n/a	99424
WLAN> up > net NET> ifconfig 192.168.77.1 255.255.255.0 NET> ping 192.168.77.10	n/a	99424
n/a	$ ath_console.exe rx 192.168.77.10 6000 udp v4	99424
NET> benchtx 192.168.77.10 6000 udp 1000 0 150 100	n/a	104240
NET> benchquit	n/a	99424
NET> up > wlan WLAN> disconnect	n/a	99424
WLAN> disable	n/a	82992

7.2.5 BLE

SX-ULPGN-BTZ EVK (Dev 1) BT Addr = 00001D123456	Heap Used (Dev 1) [byte]	SX-ULPGN-BTZ EVK (Dev 2) BT Addr = 777777777777	Heap Used (Dev 2) [byte]
After bootup	77624	After bootup	77624
> ble BLE> InitializeBluetooth BLE> SetAdvertisingParameters 60 60 1	110552	> ble BLE> InitializeBluetooth BLE> SetConnectionParameters 8 8 0	110536
BLE> AdvertiseLE 1	110584	n/a	110536
n/a	114792	BLE> ConnectLE 0 00001D123456	114768
n/a	114792	BLE> StartRxTest	114768
BLE> StartACLTxTest 777777777777 27	114792	n/a	114768
BLE> StopTxRxTest	114792	n/a	114768
n/a	110824	BLE> DisconnectLE	110808
BLE> ShutdownBluetooth	83136	BLE> ShutdownBluetooth	83128

7.2.6 IEEE 802.15.4

SX-ULPGN-BTZ EVK (Dev 1) Ext Addr = 0000FFFE00001	Heap Used (Dev 1) [byte]	SX-ULPGN-BTZ EVK (Dev 2) Ext Addr = 0000FFFE0002	Heap Used (Dev 2) [byte]
After bootup	77624	After bootup	77624
> hmi HMI> Initialize 1 1 HMI> SetPIBRequest 0 13	87048	> hmi HMI> Initialize 2 0 HMI> SetPIBRequest 0 13	87048
HMI> AddDevice 0000FFFE0002 2 0	87048	HMI> AddDevice 0000FFFE0001 1 1	87048
n/a	87048	HMI> StartReceive 1	87128
HMI> StartSend 1 1	87560	n/a	87392
HMI> StopSend 1	87048	n/a	87128
n/a	87048	HMI> StopReceive 1	87048
HMI> Shutdown	80472	HMI> Shutdown	80472