STM32F40_41xxx Bluetopia (FreeRTOS) Release Notes

Revision Control

Rev	Update	Date	Author
1.0	First Release	Feb 9,2015	R.Malovany
1.1	Update to version 4.0.2.1	Jun 17,2015	R.Malovany
1.2	Removed support for some profiles and services.	Oct 13, 2015	S. Hishmeh

Table of Content

1. Terms & Abbreviations

Abbreviation /Term	Meaning / Explanation
GAP	Generic Access Profile
HID	Human Interface Device
SPP	Serial Port Profile
A2DP	Advanced Audio Distribution Profile
AVRCP	Audio Video Remote Control Profile
GAVD	Generic Audio/Video Distribution Profile
GATT	Generic Attribute Profile
L2CAP	Logical Link Control and Adaptation Protocol
SDP	Service Discovery Protocol
HFP	Hands Free Profile
PBAP	Phonebook profile
HSP	Headset profile
CSCP	Cycling speed and cadence Profile
ANP	Alert Notification Profile
HOGP	Hid Over GATT profile
HRP	Heart Rate Profile
HSP	Headset Profile
HTP	Health Thermometer Profile
MAP	Message Access Profile
PASP	Phone Alert Status Profile
PBAP	Phone Book Profile
PXP	Proximity Profile
FMP	Find Me Profile

1. Introduction

Thank you for selecting the TI's solution.

This release adds support to the STM3240G-EVAL platform based on FreeRTOS, and includes sample applications projects using the IAR Embedded Workbench for ARM and Keil uVision IDEs. GCC compatible libraries are included as well. Please Contact Texas Instruments for any technical support issues. This release includes the core protocol stack with Low Energy support, Serial Port Profile (SPP), SPP on LE pseudo profile (SPPLE), Human Interface Device Profile (HID), Advanced Audio Distribution Profile (A2DP), Audio Video Remote Control Profile (AVRCP), Generic Audio/Video Distribution Profile (GAVD), Generic Attribute Profile (GATT) and the classic Bluetooth lower protocols, AVCTP, GAP, L2CAP, RFCOMM, SDP.

Sample applications that demonstrate the use of each of the profiles are a part of the release.

This release has been ported to TI CC256x controller only.

2. Changed in rev 1.1 (v4.0.2.1)

2.1. HW\SW updated requirements

- ST Micro STM3240G-EVAL Evaluation Board.
- TI's CC256x or CC256xB baseband (CC2564MODNEM or CC256xQFNEM).
- CC256XEM – STADAPT CC256xEM Bluetooth Adapter Kit.
  - o This adapter can be used for both EVAL and Discovery development boards.
- IAR Embedded Workbench 7.2/7.3 and up for ARM or Keil uVision 4.70.0.0 and up

2.2. HW limitation STM32 –EVAL

- **HFP\HS** – Audio is not heard well in the TX path, due to HW limitation of the STM32 EVAL platform. The microphone in the EVAL platform is connected directly to an A\D and not an actual codec that sample the microphone directly without any filter.

2.3. HW modification for PCM audio related Sample apps

The CC256x Bluetooth module comes with default HW setup.

In order to work with the EVAL platform and enable the audio related sample app. the CC256x can be only slave on the PCM.

The default HW setup is not configured to use the CC256x PCM as Slave.

The default HW setup can be used with all the Demo application that doesn't use the CC256x PCM interface as slave, like the AUDDemo application or any other logical sample apps that are not related to audio.

The Demo applications with the CC256x PCM interface in slave role are:

- HFPDemo
- HFPAGDemo
- A3DPDemo_SNK
- A3DPDemo_SRC

And in order to use those applications with audio support, HW changes need to be done:

- Place 0 OHM resistor between R18 pads.
- On the QFN EM board another change need to be done - Remove R11.

2.4. Enable codec support

There are several profiles that have audio capabilities that are using the PCM lines from the CC256X. The following sample apps have these capabilities:

- A3DPDemo_SRC
- A3DPDemo_SNK
- HSPDemo
- HFPAGDemo
- HFPDemo

Important Note:

In order to enable the audio capabilities in these specific applications please refer to HW modification for PCM audio related Sample apps paragraph 2.2 in this doc.

2.5. DMA support – Enabled by default

By default the sample application source code is loaded with **HCITRANS.C** from
Bluetopia\hcitrans.**dma** In order to work with non DMA, please change the **HCITRANS.C** to be
loaded from **Bluetopia\hcitrans** Please refer to **STM32F4xx Porting.pdf** document for more
info.

Note

Full test cycle was performed on the DMA version, and only a subset of the tests was done on
the DMA.

2.6. New profiles support

A new set of profiles support were added to this release, the following profiles support were
added:
BR\EDR

- Advanced Audio Distribution Profile
- Audio/Video Remote Control Profile
- Handsfree Profile
- Headset Profile
- Message Access profile
- Phone Book Access Profile

LE

- Alert Notification Service
- Battery Service
- Human Interface Device Service
- Heart Rate Service
- Health Thermometer Service
- Immediate Alert Service
- Link Lost Service
- Phone Alert Status Service
- TX Power Service

2.7. Enable Discovery board

This release adds support to the STM3240G-EVAL platform based on FreeRTOS, and includes sample applications projects using the IAR Embedded Workbench for ARM and Keil uVision IDes. The sample apps are written for the STM3240G-EVAL platform only, there are several changes that need to be done in order to be able to work with the samples app on discovery platform.

Note

Audio related samples apps would need an additional porting in order to enable the audio on the Discovery board; this is not a part of this release

Please refer to **Enable_STM32_Discovery.pdf** file for further instructions.

2.8. Bluetooth STACK core updates and bug fixes

Core updates

HCI

- Consider positive return values from **HCI_ReconfigureDriver** to indicate success. This is necessary as reconfiguring the COM protocols will return1 if they are in the sleep state.

L2CAP

- Verify that opcode is valid for signaling channel messages.
- Verify connectionless channel packets for minimum size
- fixed case where list structures were referenced after being freed in error handling code
- Correct the Authentication state for Security Mode 3. This corrects a connection stall that was observed using the Microsoft keyboard that contains a 1.2 radio and older stack

GAP

- Fixed memory leak in **ParseExtendedInquiryDataRecords** in where previously allocated data buffers were not freed when an allocation failed.
- Fixed **ParseExtendedInquiryDataRecords** to allow for fields with length 1 (i.e., type bytes only)
- Added code to **IssueRemoteFeaturesCallback**() in **GAP.c** to not de-reference the LMP Features pointer if it is NULL (since parts of the code pass NULL for this parameter)

GAPLE

- Fixed error code where case where device isn't connected.
- Fixed some more cases where return value was not set correctly when calling **ScheduleAsynchronousCallback**().

SDP

- Fixed improper return value from **SDPParseServiceAttributeResponseData**() internal function in **SDP.c**

- Fixed **ParseDataElements**() internal function to correctly free the allocated memory when an error occurs in parsing.

RFCOMM

- Updated RFCOMM to handle a Remote Port Negotiation for a channel that is not yet established. This was done to pass PTS test TP/RFC/BV-19-C.

Profiles updates

GATT

- Fixed bugs relating to discovering included services and dealing with included service's whose UUID is not 16 bits (and thus must be read using a separate operation).

- Fixed a bug where we returned all types of UUID (regardless of the size) in the value for an Include Definition when it was read by a remote device. According to the specification the UUID of an included service is only contained in the Include Definition's value if and only if it is a 16 bit UUID.

HFRE

- Update the Dial Number command function to passes its command argument, the dial string, as is, rather than attempting to filter valid characters. This follows GSM and ITU-T spec recommendations.

- Update **CheckStartingDigit**() to check for all valid starting digits -according to the GSM spec (as well as the '(' character - in case someone formats an area code in parenthesis for readability).

2.9. New libraries support

New set of build option were added for every profile and core libraries Multiple CPU and FPU build options:

- M3 : build option : **–cpu=Cortex-M3 ,**

- M4 : build option : **–cpu=Cortex-M4**

- M4 with HW floating point support **–cpu=Cortex-M4 --fpu=fpv4-sp**

A GCC compatible library will also be a part of every folder.

Core library will be located in **\Bluetopia\lib**, and will include the following folders and files:

- Ewarm

  - **Bluetopia.a** – this is a copy of **Bluetopia_16_M3.a** in order to simplify the Sample application linkage configuration.

  - **Bluetopia_16_M3.a -** M3 build option.

  - **Bluetopia_16_M4.a** – M4 build option.

  - **Bluetopia_16_M4.fp_HW_FP.a –** M4 HW floating point support

- GCC

  - **libBluetopia_32_M3.a -** M3 build option with **WCHAR=--wchar32**

  - **libBluetopia_32_M4.a** – M4 build option. with **WCHAR=--wchar32**

  - **libBluetopia_32_M4.fp_HW_FP.a –** M4 HW floating point support with **WCHAR=--wchar32**

- rvmdk

  - **Bluetopia.lib** – this is a copy of **Bluetopia_16_M3.a** in order to simplify the Sample application linkage configuration.

  - **Bluetopia_16_M3.lib -** M3 build option.

  - **Bluetopia_16_M4.lib** – M4 build option.

  - **Bluetopia_16_M4.fp_HW_FP.lib –** M4 HW floating point support

Every profile will also have a complete set of library support, let look at the PBAP for an example:

Located is **Bluetopia\profiles\PBAP\lib** and will include the following folders and files:

- Ewarm

- o **SS1BTPBA.a** – this is a copy of **SS1BTPBA_16_M3.a** in order to simplify the Sample application linkage configuration.
  - o **SS1BTPBA_16_M3.a** - M3 build option.
  - o **SS1BTPBA_16_M4.a** – M4 build option.
  - o **SS1BTPBA_16_M4.fp_HW_FP.a –** M4 HW floating point support

- GCC

  - o **libSS1BTPBA_32_M3.a** - M3 build option with **WCHAR=--wchar32**
  - o **libSS1BTPBA_32_M4.a** – M4 build option. with **WCHAR=--wchar32**
  - o **libSS1BTPBA_32_M4.fp_HW_FP.a –** M4 HW floating point support with **WCHAR=--wchar32**

- rvmdk

  - o **SS1BTPBA.lib** – this is a copy of **SS1BTPBA_16_M3.lib** in order to simplify the Sample application linkage configuration.
  - o **SS1BTPBA_16_M3.lib** - M3 build option.
  - o **SS1BTPBA_16_M4.lib** – M4 build option.
  - o **SS1BTPBA_16_M4.fp_HW_FP.lib –** M4 HW floating point support

2.10. Enable HW floating point library usage in IAR

PBAPDemo as an example

- Open the project located in :
  "**STM3240G-EVAL\Samples\PBAPDemo\FreeRTOS\EWARM\PBAPDemo.eww**"
- Go to project options –'right click on the PBAPDemo-Debug'→Options. And then select 'General options' category.
- Set the FPU to → VFPv4
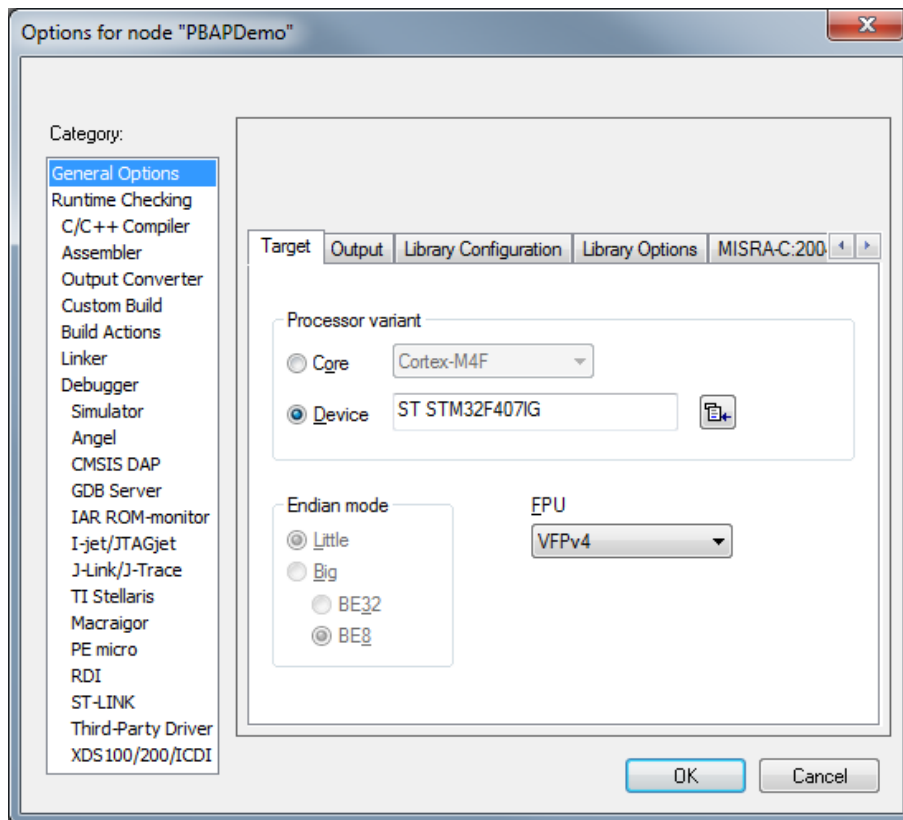
- Change the FeeeRTOS port files:
  - Open FreeRTOS folder and remove **port.c** and **portasm.s** files
  - Add **port.c** and **portasm.s** from
    '**STM3240G-EVAL\FreeRTOS\portable\IAR\ARM_CM4F**'
- Change the core and profiles library linkage inorder to link with the HW floating point pre build libraries
- Go to project options –'right click on the PBAPDemo-Debug'→Options , select 'Linker' category , choose the library tab. Remove the libraries that are currently there
  Note: we will add the same libraries location only with a different naming.
  For this example we got the following libraries that are needed for this sample app:
  Core: $PROJ_DIR$\..\..\..\..\..\Bluetopia\lib\ewarm\Bluetopia.a , we will change this to
  $PROJ_DIR$\..\..\..\..\..\Bluetopia\lib\ewarm\Bluetopia_16_M4.fp_HW_FP.a

Profile: $PROJ_DIR$\..\..\..\..\..\..\Bluetopia\\profiles\PBAP\lib\ewarm\SS1BTPBA.a ,
we will change this to:
$PROJ_DIR$\..\..\..\..\..\..\Bluetopia\\profiles\PBAP\lib\ewarm\SS1BTPBA_16_M4.fp_HW_FP.a
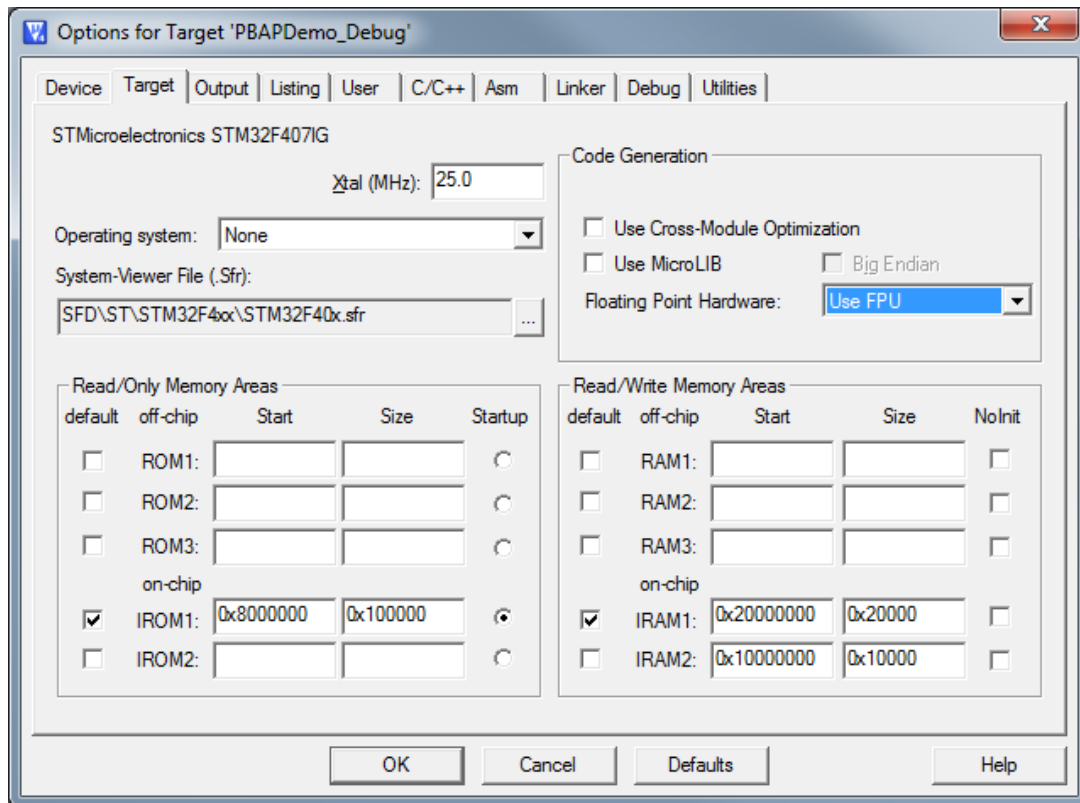
It should look like this:
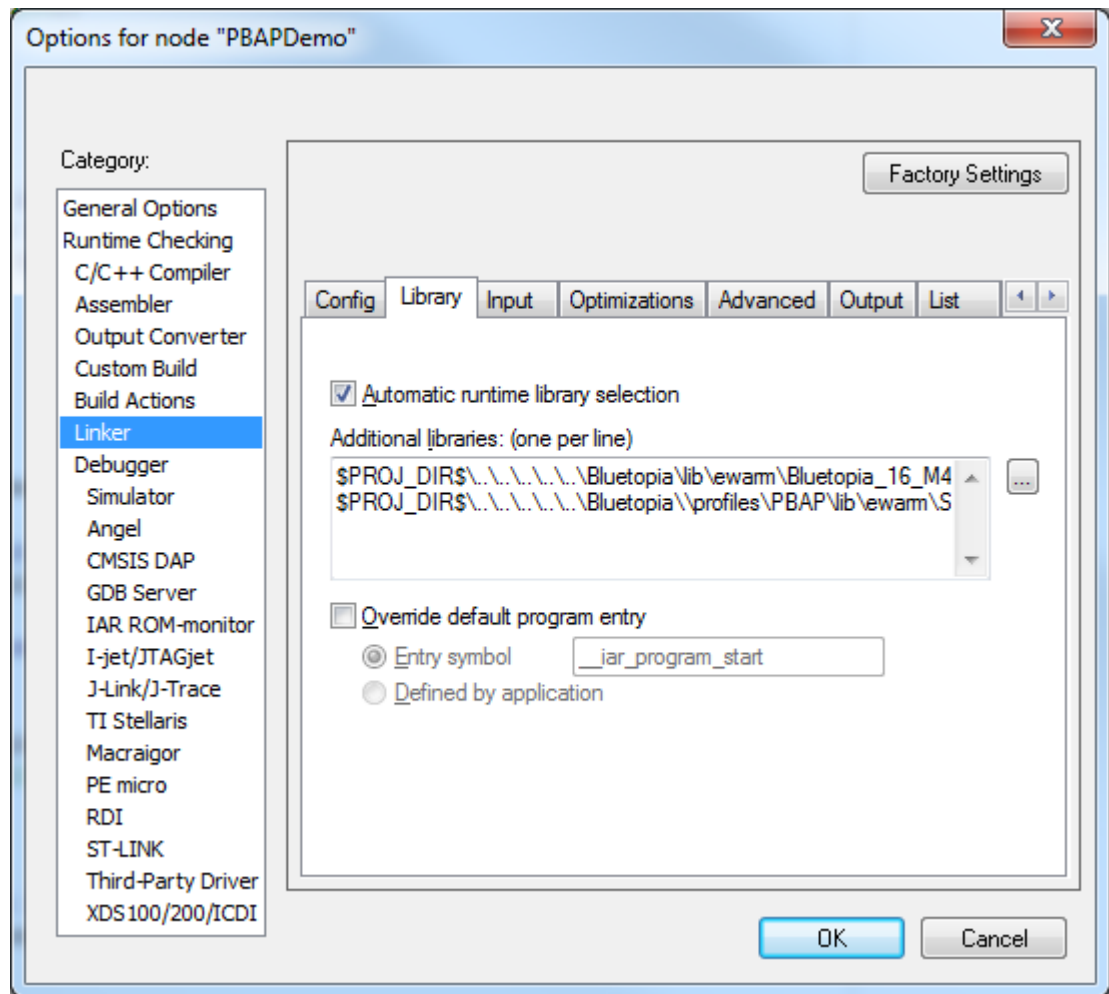
- Preform Clean and rebuild the entire project.

2.11. Enable HW floating point library usage in KEIL

PBAPDemo as an example

- Open the project located in :
  "**STM3240G-EVAL\Samples\PBAPDemo\FreeRTOS\RVMDK\PBAPDemo.uvproj**"
- Go to project options –'right click on the PBAPDemo-Debug'→Options for target.
- In the code generation change the 'Floating point Hardware' to Use FPU and press OK.

- At the project folders expend **FreeRTOS** folder and remove **port.c** file.
- Right click on the **FreeRTOS** folder and chose '**add files to group FreeRTOS'** and add the following file:
  'STM3240G-EVAL\FreeRTOS\portable\RVDS\ARM_CM4F\port.c'

- At the project folders remove the **.lib** files from the folders.
- At the HW floating point enabled libraries crospanding to this project in this demo is the PBAP lib and the core lib. Right click the libraries folder and chose '**add files to group Libraries'** , add the following files:

  - **Bluetopia\lib\rvmdk\Bluetopia_16_M4.fp_HW_FP.lib**
  - **Bluetopia\profiles\PBAP\lib\rvmdk\SS1BTPBA_16_M4.fp_HW_FP.lib**

- Preform rebuild to the project

2.12. New set of API Documentation

- Alert Notification Service
- Battery Service
- Device Information Service
- Headset Profile
- Health Thermometer Service
- Heart Rate Service
- HID Host Subsystem
- Immediate Alert Service
- Link Loss Service
- Message Access Profile
- Phone Alert State Service
- Phonebook Access Profile
- Tx Power Service

2.13. New Sample apps support

A3DPDemo_SRC

This application shows how to utilize the A2DP function of the Bluetopia core stack and the Assisted A2DP (A3DP) of the TI CC2564. This will act as a Source device and also demonstrates the AVRCP Target role.

A3DPDemo_SNK

This application shows how to utilize the A2DP function of the Bluetopia core stack and the Assisted A2DP (A3DP) of the TI CC2564. This will act as a Sink device and also demonstrates the AVRCP Controller role

ANPDemp

This application shows how to utilize Low Energy (LE) and the GATT profile. This sample implements the Alert Notification Profile (ANP) using the GATT profile. This sample acts as a LE Master and LE Slave.

FMPDemo

This application shows how to utilize Low Energy (LE) and the GATT profile. This sample implements the Find Me Profile (FMP) using the GATT profile. This sample acts as a LE Master and LE Slave.

HOGPDemo

This application shows how to utilize Low Energy (LE) and the GATT profile. This sample implements the HID over GATT Profile (HOGP) using the GATT profile This sample acts as a HID device (LE Master).

HRPDemo

This application shows how to utilize Low Energy (LE) and the GATT profile and Implements the Heart Rate Monitoring Profile (HRP) using the GATT profile. This sample acts as a LE Master and LE Slave.

HSPDemo

This application shows how to utilize the HSP (Headset Profile) module, and how to handle the different HSP callback events. This will act only as AG and HS roles

HTPDemo

This application shows how to utilize Low Energy (LE) and the GATT profile and implements the Health Thermometer Profile (HTP) using the GATT profile. This sample acts as a LE Master and LE Slave.

MAPDemo

This application shows how to utilize the MAP (Message Access Profile) module and how to handle the different MAP callback events. This will act as either a MAP server or a MAP client.

PASPDemo

This application shows how to utilize Low Energy (LE) and the GATT profile and implements the Phone Alert Status Profile (PASP) using the GATT profile. This sample acts as a LE Master and LE Slave.

PBAPDemo

This application shows how to utilize the PBAP (Phonebook Access Profile) module and how to handle the different PBAP callback events. This will act as either a PBAP server or a PBAP client

iBEACONDemo

This application shows how to utilize Low Energy (LE) and advertise the parameters according to Apple iBEACON spec.

PXPDemo

This application shows how to utilize Low Energy (LE) and the GATT profile. This sample implements the Proximity Profile (PXP) using the GATT profile. This sample acts as a LE Master and LE Slave.

2.14. Update FBL (Flexible build library) options

FBL options were updated to have 3 different sets of libraries:

- M3
- M4
- M4 with HW floating point support.

Please refer to **Using the Flexible Build Library.pdf** document for more info.

2.15. HFP 3 way calling support

- HFPDemo application was updated to support 2 phones in parallel
- Add audio routing to the STM3240G-EVAL board- The CC256x Bluetooth module and the on-board DAC are the PCM slaves and the STM32 is the PCM master that provides the clock and the FSynch lines. The input audio from the Bluetooth configured to come from the CC256x PCM directly to the DAC. The input MIC from wired headset, that connect to the EVAL board, is sampled via the ADC peripheral of the STM32 and then output to the PCM for the CC256x and then to the Bluetooth transmission. For all the above a driver to the STM32 I2S and driver to the DAC CS43l22 added
- Add two phones handling with the ability to open two HFP servers with structure for each one to hold the status. Each phone can also open SCO/eSCO but the audio always routed to the first one that opened.
- Add user function, **SwitchAudioFor2PhonesWithActiveCall**(), to switch the audio between two active calls in the two phones.
- Add user function **AnswerSeconePhoneIncomingCallAndSwitchAudio**(), to answer incoming call from second phone during active call in the first call and rout the audio to the second phone

2.16. Removed CSCP support

CSCP was removed till we will have it qualified by the SIG.

3. Changed in rev 1.2 (v4.0.2.2)

Removed support for the following profiles and services:

- Device ID Profile
- File Transfer Profile
- Health Device Profile
- Object Push Profile
- Personal Area Networking
- Current Time Service
- Glucose Service
- Next DST Change Service
- Reference Time Update Service
- Scan Parameters Service

4. Overview

This release includes support for the STM32F407IG processor. The following precompiled libraries, which are part of this release, were built using the ARM compiler 5.04:

- **Bluetopia.lib**, **Bluetopia.a** – Core Bluetopia library (HCI, L2CAP, SDP, RFCOMM, SPP, GAP)
    - **Bluetopia.a** – Core for IAR Embedded Workbench
    - **Bluetopia.lib** – Core for Keil uVision
    - **libBluetopia.a** – GCC compatible library

The following source files, which are part of this release, were ported to STM32F407IG:

- **BTPSKRNL.c** – O/S abstraction layer
- **HCITRANS.c** –HCI UART transport abstraction layer

Porting documentation for the required functions is included for reference.

Finally, this release contains sample applications in source format. Those applications are fully functional applications that show the following usage:

- Initializing/shutting down of the stack

- Device Discovery (Inquiry and SDP)

- Device Pairing

- Serial Port Profile (SPP) usage

- Advanced Audio Distribution (A2DP) Profile sink usage

- Human Interface Device (HID) usage

- Hands Free Profile (HFP) usage

- LE usage – demonstrates through a dedicated proprietary SPP like service

- Local Device Management (class of device, local name, device address, connectability modes, discoverability modes, pairability modes, etc).

The sample applications provide a command line interface (through a UART or some other mechanism) that can be used to issue commands in real-time and watch the results. These samples should easily be modifiable to add/change features during testing/development.

5. Documentation

- API documentation
  - o Advanced audio distribution profile
  - o Audio profiles subsystem
  - o Audio video remote control profile
  - o Bluetopia core API
  - o Generic access profile service
  - o Generic attribute Profile
  - o Generic audio video distribution profile
  - o Hands free profile
  - o Human interface device profile
  - o Human interface device service
  - o Sub band SW CODEC

- Porting documentation
    - Architecture overview
    - System call requirements
    - BTPSKRNL (O/S abstraction layer)
    - HCITRANS (HCI Transport abstraction layer)
- Memory analysis report
    - CC256x STM32 FreeRTOS Bluetopia SDK v4.0.2.0 Memory Analysis Report
- STM32F40_41xxx porting
- Using the Flexible Build Library

All documentation is located under the Documentation sub directory.

6. Hardware/Software Requirements

- ST Micro STM3240G-EVAL Evaluation Board.
- TI's CC256x or CC256xB baseband (CC2564MODNEM or CC256xQFNEM).
- Interface board, CircuitCO, to connect between the TI's Bluetooth CC256x or CC256xB board and the STM3240G-EVAL board (999-0003923).
- IAR Embedded Workbench 7.2/7.3 for ARM or Keil uVision 4.70.0.0.

7. Hardware Jumper Requirements

The following jumper settings are required to operate the sample applications on the STM3240G-EVAL board. The jumpers are on the CircuitCO interface board.

- Non DMA
    - JP2 – JP5, Set to the UART5 position.
- DMA
    - JP2 – JP5, Set to the UART6 position (please refer to DMA support in this document).
- Close nSHUTD with a jumper\wire.

8. Arm compiler library notes

The libraries contained in this distribution were compiled using the ARM compiler 5.04. Users simply are required to link with the Bluetopia.lib library or Bluetopia. A for core stack functionality (stored in the folder lib). All other profiles and support libraries have a corresponding library directory (also named lib) which contains the library required for linking

9. General Porting Notes

Porting Bluetopia stack for a new platform includes the following steps. Further Information on the various steps will be provided on next sections:

- Port **BTPSRKNL.c** to support the target platform (provided)
- Port **HCITRANS.c** to support the target platform (provided)
- Port **BTPSVEND.c** to support the target TI Bluetooth chipset (provided)
- Create the sample/test application
- Create a project that includes the above ported files, the sample/test application, and links with supplied binary libraries.

9.1. BTPSKRNL Porting Notes

For information on porting **BTPSKRNL.c** to the target platform, please see the attached BTPSKRNL porting documentation.

9.2. HCITRANS Porting Notes

For information on porting **HCITRANS.c** to the target platform, please see the attached HCITRANS porting documentation. The **HCITRANS.c** file located in the Core directory should be used (Multi-threaded version).

.

9.3. BTPSVEND Porting Notes

This module is primarily to configure the specific Bluetooth chipset. This typically entails:

- Required chipset Patch RAM
- Required chipset default Bluetooth values (Name, Class of Device, etc)
- Radio Tuning parameters (specific to physical board layout)
- Means to apply a unique BD_ADDR to the chipset

The content of what is actually downloaded to the chip is typically provided by TI. The implementation of how to download these configuration parameters is what is contained in this module.

10. DMA Support

By default the sample application source code is loaded with **HCITRANS.C** from **Bluetopia\hcitrans**. In order to work with DMA, please change the **HCITRANS.C** to be loaded from **Bluetopia\hcitrans.dma.** Please refer to **STM32F4xx Porting.pdf** document for more info.

Note: Full test cycle was performed on the non-DMA version, and only a subset of the tests was done on the DMA.

11. Flexible Build Library

The Flexible Build Library (FBL) is a mechanism that allows the user to generate different versions of the Bluetopia library tailored to the specified use case. This allows the user to reduce the flash memory requirements of the Bluetopia stack by removing code that is not needed by the user's specific application. Some examples are adding or removing Low Energy support, adding or removing SPP support and adding or removing SCO audio support for more info please refer to **Using the Flexible Build Library.pdf** document

12. Application Development Notes

This section details specific usage information on how to effectively use Bluetopia with C/C++ Compilers.

In order to include all necessary prototypes/constants, the programmer needs to only include the SS1BTPS.h header file (for the core library). In order to include all necessary prototypes/constants for profiles, the programmer needs to only include the SS1BTxxx.h for the specific profile (where, 'xxx" is the profile name).

Users using C++ MUST use the following code to include the SS1BTPS.H Header File in their C++ Source:

```
extern "C" {
#include "SS1BTPS.h"
};
```

Users using C (and NOT C++) do NOT need to include the above code snippet and can simply include the SS1BTPS.h Header File as normal, for example:

```
#include "SS1BTPS.h"
```

All API structures that are used with Bluetopia at the hardware device interface level are aligned on default packing boundaries. This shouldn't pose much of a problem because most of the structures are aligned such that they would not cause too many packing problems otherwise. Lower level structures are aligned based on compiler switches as specified by the customer.

13. Sample Application Notes

The sample applications provided demonstrate how to utilize the various Bluetopia Application Programming Interfaces provided for each profile (GAP, SPP, GAPS, GATT, A2DP, AVRCP, HID and HF). This release is intended to be built with either IAR's Embedded Workbench for ARM or Keil uVision.

- IAR Project is included in the **\[Demo app name]\FreeRTOS\EWARM**
- Keil uVision project is included in the **\[Demo app name]\FreeRTOS\RMVDK**

The sample applications can be found under the following subdirectory "Samples":

13.1. AUDDemo

This application shows how to utilize Audio over A2DP profile with SW CODEC for SBC decoding. The Application also shows how to handle the AVRCP profile for controlling the audio stream and open/close the stream with Generic Audio/Video Distribution Profile using the Audio/Video Distribution Transport Protocol (AVDTP)

13.2. HFPAGDemo

This application shows how to utilize the HFP (Hands-Free Profile) module, and how to handle the different HFP callback events. This will act only as AG role

13.3. HFPDemo

This application shows how to utilize the HFP (Hands-Free Profile) module, and how to handle the different HFP callback events. This will act only as Headset role

13.4. HIDDemo

This application shows how to utilize the HID module, and how to handle the

different HID callback events. This will act as either a HID Host or HID Device.

13.5. SPPDemo

This application shows how to utilize the SPP module, and how to handle the different SPP callback events. This will act as either a SPP Server or SPP Client.

13.6. SPPLEDemo

This application shows how to utilize Low Energy (LE) and the GATT profile as well as the Serial Port Profile. This sample emulates using SPP over LE using the GATT Profile. A device can connect to any device running this application (using LE or BR/EDR) and send data (using SPP or SPPLE server) to it. This will act as a LE Master and LE Slave. It can also act as a SPP Server and Client.

This application also shows how to utilize the SPP module and how to handle the different SPP callback events. This will act as either a SPP Server or SPP Client and exposes the same command set for SPP as the SPPDemo.

The above mentioned sample should serve as a good starting point because each Bluetooth task is broken up into a discreet function. The sample also provides the asynchronous event handlers and more importantly, the code that is required to be executed in response to the asynchronous events.

These sample applications are simple, command line applications which will display a list of available commands.

Input and Output can be utilized by plugging in the USART DB9 (CN16) to a PC via a NULL modem cable. The parameters for this port are:

Baud: 115,200

Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

Local Echo: Off