

ARM® Cordio Stack

ARM-EPM-115881 2.0

Stack System Architecture

Confidential

ARM®

ARM® Cordio Stack

System Architecture Reference

Copyright © 2012-2016 ARM. All rights reserved.

Release Information

The following changes have been made to this book:

Document History

Date	Issue	Confidentiality	Change
25 September 2015	-	Non-Confidential	First Wicentric release for 1.1 as 2012-0023
1 March 2016	A	Confidential	First ARM release for 1.1
24 August 2016	1.0	Confidential	AUSPEX # / Directory Structure
15 December 2016	1.1	Confidential	BT 5.0 Update
10 August 2017	2.0	Confidential	Folder Structure

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information: (i) for the purposes of determining whether implementations infringe any third party patents; (ii) for developing technology or products which avoid any of ARM's intellectual property; or (iii) as a reference for modifying existing patents or patent applications or creating any continuation, continuation in part, or extension of existing patents or patent applications; or (iv) for generating data for publication or disclosure to third parties, which compares the performance or functionality of the ARM technology described in this document with any other products created by you or a third party, without obtaining ARM's prior written consent.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to ARM's customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow ARM's trademark usage guidelines at <http://www.arm.com/about/trademark-usage-guidelines.php>

Where the term ARM is used it means "ARM or any of its subsidiaries as appropriate".

Copyright © 2012-2016, ARM Limited or its affiliates. All rights reserved.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20348

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

Contents

ARM® Cordio Stack	1
1 Preface	6
1.1 <i>About this book</i>	6
1.1.1 <i>Using this book</i>	6
1.1.2 <i>Intended audience</i>	6
1.1.3 <i>Terms and abbreviations</i>	6
1.1.4 <i>Conventions</i>	8
1.1.5 <i>Additional reading</i>	8
1.2 <i>Feedback</i>	8
1.2.1 <i>Feedback on content</i>	9
2 Introduction	10
3 Software System	11
3.1 <i>System Configuration</i>	11
3.2 <i>Cordio Profiles</i>	11
3.2.1 <i>Sample Applications</i>	12
3.2.2 <i>Profiles and Services</i>	12
3.2.3 <i>App Framework</i>	12
3.3 <i>Cordio Stack</i>	13
3.3.1 <i>ATT</i>	13
3.3.2 <i>SMP</i>	13
3.3.3 <i>L2C</i>	13
3.3.4 <i>HCI</i>	14
3.3.5 <i>DM</i>	14
3.4 <i>WSF</i>	14
4 Software Architecture	15

4.1	<i>Interface Architecture</i>	15
4.1.1	<i>Message Passing API Functions</i>	15
4.1.2	<i>Direct Execute API Functions</i>	15
4.1.3	<i>Callback Functions</i>	15
4.2	<i>Event Handlers and Tasks</i>	15
5	<i>Data Path</i>	17
5.1	<i>TX Path</i>	17
5.2	<i>RX Path</i>	17
6	<i>Porting</i>	19
6.1	<i>WSF Porting</i>	19
6.2	<i>HCI Porting</i>	19
7	<i>Directory Structure</i>	20
7.1	<i>ble-host directory</i>	20
7.2	<i>ble-profiles directory</i>	20

1 Preface

This document describes the ARM Cordio software system and lists the API functions and their parameters.

1.1 About this book

This book is written for experienced software engineers who might or might not have experience with ARM products. Such engineers typically have experience of writing Bluetooth applications but might have limited experience of the Cordio software stack.

It is also assumed that the readers have access to all necessary tools.

1.1.1 Using this book

This book is organized into the following chapters:

- **Introduction**
Read this for an overview of the process of configuring and implementing the IoT subsystem.
- **Software System**
Read this for an overview of components in the software.
- **Software Architecture**
Read this for a description of the functions in the API.
- **Data Path**
Read this for a description of the data flow from the application, stack, and the HCI.
- **Porting**
Read this for a description of the porting process.
- **Directory Structure**
Read this for the details of the directories.
- **Revisions**
Read this chapter for descriptions of the changes between document versions.

1.1.2 Intended audience

This book is written for experienced hardware and *System-on-Chip* (SoC) engineers who might or might not have experience with ARM products. Such engineers typically have experience of writing Verilog and of performing synthesis, but might have limited experience of integrating and implementing ARM products.

It is also assumed that the readers have access to all necessary tools.

1.1.3 Terms and abbreviations

For a list of ARM terms, see the ARM [glossary](#).

Terms specific to the Cordio software are listed below:

Term	Description
ACL	Asynchronous Connectionless data packet
AD	Advertising Data
ARQ	Automatic Repeat reQuest
ATT	Attribute Protocol, also attribute protocol software subsystem

ATTC	Attribute Protocol Client software subsystem
ATTS	Attribute Protocol Server software subsystem
CCC or CCCD	Client Characteristic Configuration Descriptor
CID	Connection Identifier
CSRK	Connection Signature Resolving Key
DM	Device Manager software subsystem
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HCI	Host Controller Interface
IRK	Identity Resolving Key
JIT	Just In Time
L2C	L2CAP software subsystem
L2CAP	Logical Link Control Adaptation Protocol
LE	(Bluetooth) Low Energy
LL	Link Layer
LLPC	Link Layer Control Protocol
LTK	Long Term Key
MITM	Man In The Middle pairing (authenticated pairing)
OOB	Out Of Band data
SMP	Security Manager Protocol, also security manager protocol software subsystem
SMPI	Security Manager Protocol Initiator software subsystem
SMPR	Security Manager Protocol Responder software subsystem
STK	Short Term Key
WSF	Wireless Software Foundation software service and porting layer.

1.1.4 Conventions

The following table describes the typographical conventions:

Typographical conventions	
Style	Purpose
<i>Italic</i>	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
MONOSPACE	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u>MONOSPACE</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
monospace <i>italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
monospace bold	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>ARM[®] Glossary</i> . For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

1.1.5 Additional reading

This section lists publications by ARM and by third parties.

See [Infocenter](#) for access to ARM documentation.

Other publications

This section lists relevant documents published by third parties:

- Bluetooth SIG, “*Specification of the Bluetooth System*”, Version 4.2, December 2, 2015.
- Bluetooth SIG, “*Specification of the Bluetooth System*”, Version 5.0, December 7, 2016.

1.2 Feedback

ARM welcomes feedback on this product and its documentation.

1.2.1 Feedback on content

If you have comments on content then send an e-mail to support-cordio-sw@arm.com. Give:

- The title.
- The number, ARM-EPM-115881.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note: ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

2 Introduction

This document describes the system architecture of ARM's Bluetooth low energy software system.

3 Software System

The Bluetooth LE software system consists of two main components:

- The Cordio Stack is complete protocol stack solution for single-mode Bluetooth LE devices.
- The Cordio Profiles consists of sample applications, interoperable Bluetooth profile and service components, and a service layer for simplified application development and porting.

The software system is built on the Wireless Software Foundation (WSF), an OS wrapper and porting layer. WSF also provides general-purpose software services such as queues, timers, and buffer management.

3.1 System Configuration

The *Cordio* Stack and Profiles are designed to support single-chip SoC systems and dual-chip systems.

When operating in a single-chip system the *Cordio* Stack and Profiles run on the processor inside the SoC. A "thin" HCI layer adapts to the software interface of the target's LE link layer.

When operating in a dual-chip system the *Cordio* Stack and Profiles run on a microcontroller and communicate with a Bluetooth LE controller chip over a wired interface such as UART or SPI. A standard transport-based HCI layer manages the communication between the two devices.



Figure 1. *Cordio* Stack and Profiles in a single-chip SoC system and dual-chip system

3.2 *Cordio* Profiles

ARM's *Cordio* Profiles consist of sample applications, interoperable Bluetooth profile and service components, and a service layer called the App Framework for simplified application development and porting.

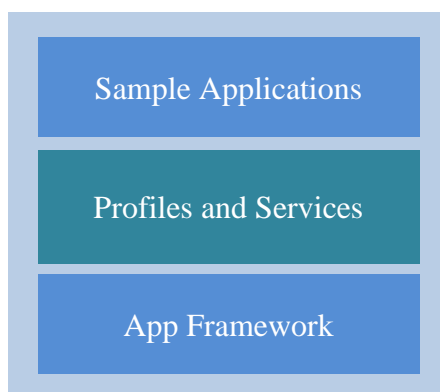


Figure 2. Cordio Profiles software system.

3.2.1 Sample Applications

ARM's Bluetooth low energy sample applications provide example source code for products such as a proximity keyfob, health sensor, and watch. The sample applications are designed with a product-oriented focus, with each application supporting one or more Bluetooth LE profile. The sample applications interface to the Profiles and Services and the App Framework.

3.2.2 Profiles and Services

The profiles and services are interoperable components designed to Bluetooth profile and service specification requirements. The profiles and services are used in applications to implement particular profile and service features.

The profiles are implemented in separate files for each profile role. The services, however, may be grouped together in files based on their logical function and the profile they are used by.

3.2.3 App Framework

The App Framework performs many operations common to Bluetooth LE embedded applications, such as:

- Application-level device, connection, and security management.
- Simple user interface abstractions for button press handling, sounds, display, and other user feedback.
- An abstracted device database for storing bonding data and other device parameters.

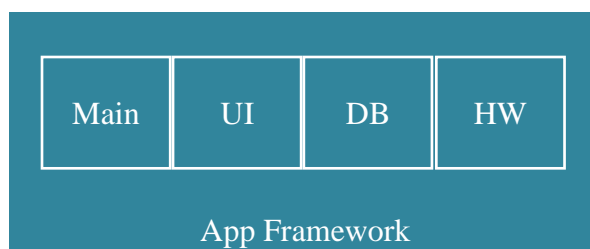


Figure 3. App Framework software subsystem

The App Framework consists of several modules, each with their own API interface file:

- **Main:** Device, connection, and security management.
- **UI:** User interface abstraction.
- **DB:** Device database.

- **HW:** Hardware sensor interface abstraction.

3.3 Cordio Stack

The *Cordio* Stack is complete host protocol stack solution for single-mode Bluetooth LE devices. It consists of five protocol layers:

- **ATT:** Attribute protocol.
- **SMP:** Security manager protocol.
- **L2C:** L2CAP protocol.
- **HCI:** Host controller interface protocol.
- **DM:** Device manager.

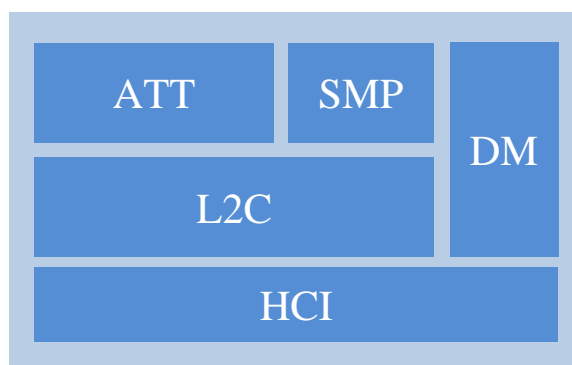


Figure 4. Cordio Stack software system

3.3.1 ATT

The ATT subsystem implements the attribute protocol and generic attribute profile (GATT). It contains two independent subsystems: The attribute protocol client (ATTC) and attribute protocol server (ATTS).

ATTC implements all attribute protocol client features and is designed to meet the client requirements of the generic attribute profile. ATTC can support multiple simultaneous connections to different servers.

ATTS implements all attribute protocol server features and has support for multiple simultaneous client connections. ATTS also implements the server features defined by the generic attribute profile.

3.3.2 SMP

The SMP subsystem implements the security manager protocol. It contains two independent subsystems:

- The initiator (SMPI). SMPI implements the initiator features of the security manager protocol and has support for multiple simultaneous connections.
- The responder (SMPR). SMPR implements the responder features of the security manager protocol and has support for only one connection (by Bluetooth specification design).

SMP also implements the cryptographic toolbox, which uses AES. The interface to AES is asynchronous and abstracted through WSF. SMP also implements functions to support data signing.

3.3.3 L2C

The L2C subsystem implements the LE L2CAP protocol. It is a substantially scaled-down version of

regular Bluetooth L2CAP.

In the TX data path, the main function of L2C is building L2CAP packets and sending them to HCI. In the RX data path, its main function is receiving packets from HCI and routing them to either SMP or ATT.

L2C also implements the connection parameter update procedure.

3.3.4 HCI

The HCI subsystem implements the host-controller interface specification. This specification defines commands, events, and data packets sent between a Bluetooth LE protocol stack on a host and a link layer on a controller.

The HCI API is optimized to be a thin interface layer for a single chip system. It is configurable for either a single chip system or traditional system with wired HCI.

This configurability is accomplished through a layered implementation. A core layer can be configured for either a single chip system or wired HCI. A transport and driver layer below the core layer can be configured for different wired transports such as UART.

3.3.5 DM

The DM subsystem implements device management procedures required by the stack. These procedures are partitioned by procedure category and device role (master or slave). The following procedures are implemented in DM:

- Advertising and device visibility: Enable/disable advertising, set advertising parameters and data, set connectability and discoverability.
- Scanning and device discovery: Start/stop scanning, set scan parameters, advertising reports, name discovery.
- Connection management: Create/accept/remove connections, set/update connection parameters, read RSSI.
- Security management: Bonding, storage of security parameters, authentication, encryption, authorization, random address management.
- Local device management: Initialization and reset, set local parameters, vendor-specific commands.

DM procedures support the Generic Access Profile (GAP) when applicable.

3.4 WSF

The Wireless Software Foundation (WSF) is a simple OS wrapper, porting layer, and general-purpose software service used by the software system. The goal of WSF is to stay small and lean, supporting only the basic services required by the stack. It consists of the following:

- Event handler service with event and message passing.
- Timer service.
- Queue and buffer management service.
- Portable data types.
- Critical sections and task locking.
- Trace and assert diagnostic services.
- Security interfaces for encryption and random number generation.

4 Software Architecture

This section describes the functions in the API.

4.1 Interface Architecture

The software system uses function calls and callback functions in its APIs, as described below.

4.1.1 Message Passing API Functions

Message passing API functions result in a message being sent to the task running the stack. These functions typically involve a complex operation, such as creating a connection, and do not access internal (private) data.

4.1.2 Direct Execute API Functions

Direct execute API functions run entirely in the context of the calling function. These functions typically involve simple operations like reading or setting internal data. Task scheduling must be locked when accessing internal data.

4.1.3 Callback Functions

Callback functions are implemented by the client using the protocol stack and execute in the context of the stack. Callback functions are used to send events and data to the client.

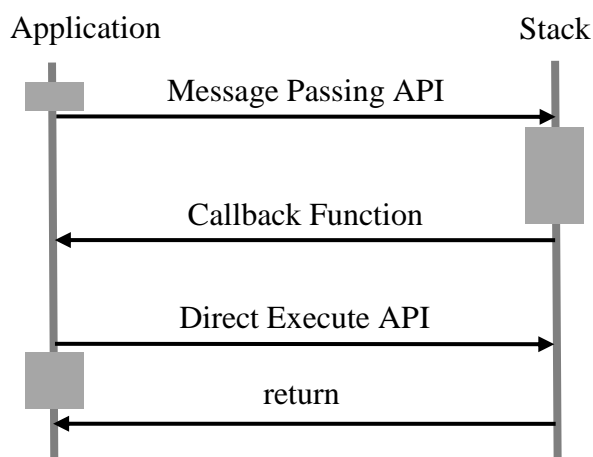


Figure 5. Message passing and direct execute interfaces.

4.2 Event Handlers and Tasks

The ARM software system defines an event handler service that forms a basis for the asynchronous communication mechanisms used in the system. An event handler can receive messages and events. Each software subsystem typically has its own event handler; for example, each layer of the protocol stack has its own event handler.

The stack is designed to be flexible and allow for different task architectures. The software system does not define any tasks but defines some interfaces to tasks. It relies on the target OS to implement tasks and manage the timer and event handler services from target OS tasks. A typical single-chip software system will use separate tasks for the application, stack, and link layer. However there is

nothing in the design of the protocol stack or profiles that prevent them from being run in the same task as other software systems.

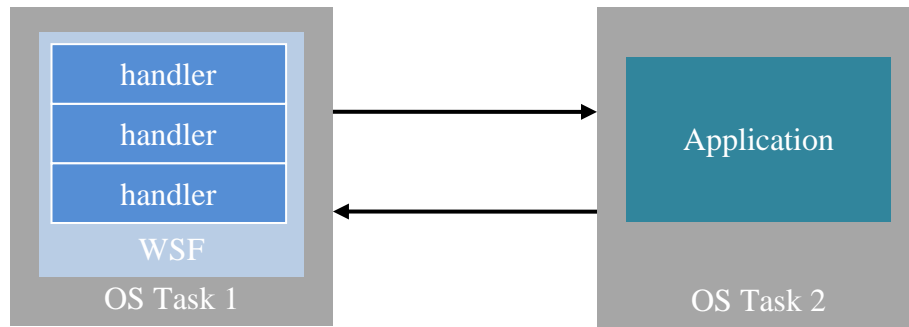


Figure 6. Example system showing event handlers executing within an OS task

5 Data Path

This section describes the data flow between applications and the HCI.

5.1 TX Path

The TX data path covers the flow of data as it is sent from the application, through the stack, and then on to HCI.

There can be two data copies in the TX data path:

- When data is sent from the application to the stack
- When data is sent from the stack to HCI.

The stack does not copy data internally between layers.

The allocation and deallocation of data buffers takes place at the point where data is copied. When the application sends data to the stack, a buffer is allocated and data is copied to the buffer. When data is sent from the stack to the HCI or the link layer, the data is copied to an HCI or link layer buffer and the stack buffer is deallocated.

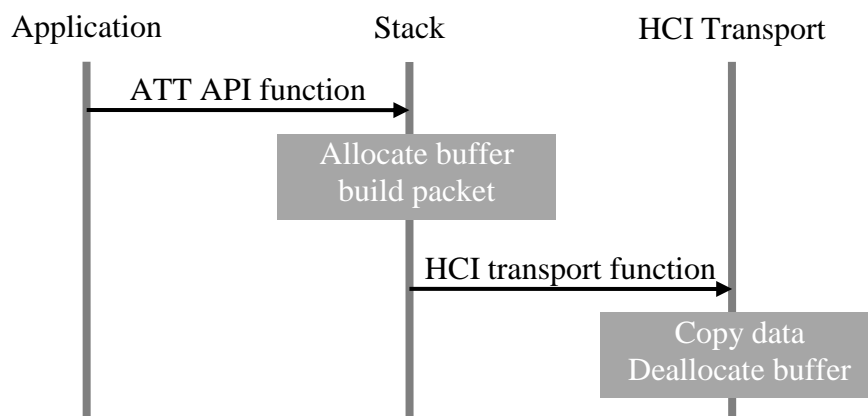


Figure 7. TX data path

5.2 RX Path

The RX data path covers the flow of data as it is sent from HCI, through the stack, and then on to the application. Like the TX path, there can be two data copies in the RX data path: when data is sent from the stack to the application and when data is sent from HCI to the stack. The stack does not copy data internally between layers.

Buffers are allocated by the HCI layer and then deallocated internally by the stack.

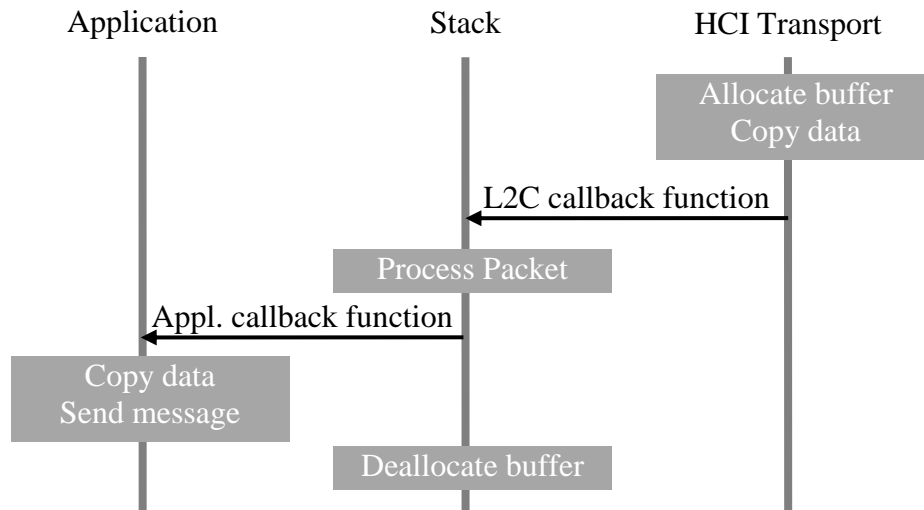


Figure 8. RX data path

6 Porting

The porting process typically consists of two main steps:

1. Porting WSF interfaces and services to the target OS and software system.
2. Porting HCI to the target system and writing a transport driver, if applicable.

6.1 WSF Porting

Porting WSF typically consists of the following steps:

1. Create common data types for the target compiler.
2. Interface to a system timer to receive timer updates.
3. Implement WSF OS wrapper functions and interfaces.
4. Implement WSF diagnostics.

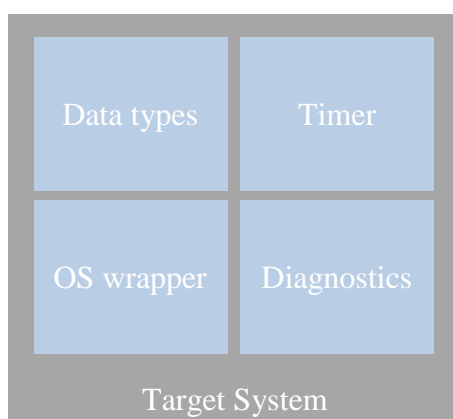


Figure 9. WSF porting process to a target system

6.2 HCI Porting

Cordio's HCI layer is designed to be portable and support different transport and chip configurations. The porting process depends on the chip configuration: If the stack is ported to a single-chip system then a "thin HCI" porting process is used. If the stack is ported to a two-chip system with wired HCI transport then a transport-based porting process is used.

7 Directory Structure

The contents of the root directory are listed in the table below:

Table 1: Root directory

Directory	Description
documentation	Documentation
wsf	Wireless Software Foundation
ble-host	Cordio Stack software
ble-profiles	Cordio Profiles platform and sample projects
platform	Platform integration and example source
projects	Cordio sample applications

7.1 ble-host directory

Table 2: sw directory

Directory	Description
build	Build configuration / Makefiles
include	Host API
sources/hci	Host HCI source
sources/sec	Host security support (AES, ECC)
sources/stack	Host stack source

7.2 ble-profiles directory

Table 3: sw directory

Directory	Description
build	Build configuration / Makefiles
include	Profiles API
sources/apps	Application framework and sample applications
sources/profiles	Bluetooth LE profiles
sources/services	Bluetooth LE services

The apps directory contains the Application Framework and sample applications.

Table 4: apps directory

Directory	Description
app	App Framework
cycling	Cycling sensor sample application
datc	Proprietary data transfer client sample application
datc	Proprietary data transfer server sample application
fit	Fitness sensor sample application
gluc	Glucose sensor sample application
hidapp	HID sample application
medc	Health data collector sample application
meds	Health sensor sample application
sensor	Sensor sample application
tag	Proximity tag sample application
uribeacon	Uribeacon sample application
watch	Watch sample application
wdxs	Wireless data exchange application

The profiles directory contains the Bluetooth LE profiles.

Table 3: profiles directory

Directory	Description
anpc	Alert Notification Profile client
bas	Battery Service server
blpc	Blood Pressure Profile client
blps	Blood Pressure Profile server
cpp	Cycling Power Profile server
cscp	Cycling Speed and Cadence Profile server
dis	Device Information Service client
fmp1	Find Me Profile locator

gap	Gap Profile
gatt	Generic Attribute Profile client
glpc	Glucose Profile client
glps	Glucose Profile server
hid	HID device
hrpc	Heart Rate Profile client
hrps	Heart Rate Profile server
htpc	Health Thermometer Profile client
htps	Health Thermometer Profile server
paspc	Phone Alert Status Profile client
plxpc	Pulse Oximeter Profile collector
plxps	Pulse Oximeter Profile sensor
rscp	Running Speed and Cadence Profile sensor
scpps	Scan Parameter Profile server
sensor	Example Temperature and Gyroscope Service Profile
tipc	Time Profile client
udsc	User Data Service Collector
uribeacon	Uribeacon Configuration Profile
wdxs	Proprietary Data Exchange Server Profile
wdxc	Proprietary Data Exchange Client Profile
wpc	Cordio proprietary profile client
wspc	Weight Scale Profile client
wsp	Weight Scale Profile server