

Project Ara

Module Developers Kit (MDK)

Release 0.10 (alpha)

April 9, 2014

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

Welcome. What now?

Project Ara is a platform for creating modular smartphones. Users will be able to populate an endoskeleton, the structural frame and network backbone of the device, and populate it with modules, the building blocks that make up the vast majority of the phone's functionality and features. Since modules are interchangeable, a user has the freedom to design exactly the phone they want and continue to customize the phone over time by replacing modules. Ara's success is predicated on a rich, vibrant, and diverse ecosystem of modules from a myriad of developers. Users would be able to select modules from an online marketplace using a configurator that facilitates user choice and curates the configuration process to ensure that the selection of modules provides the expected system-level functionality.

This document, which serves as the narrative and core of the Module Developers Kit (MDK), discusses how to develop modules compatible with the Ara platform. Note that while this document extensively describes the endoskeleton, or "endo," it does so exclusively for the benefit of module developers to ensure compatibility between modules and the endo. The Ara platform does not presently support third-party endo developers.

Chapter 1 provides a brief description of the licensing landscape that governs the Ara ecosystem. Chapter 2 talks about the industrial design of the Ara platform including the physical layout of the frame, the endoskeleton or "endo", and how modules fit within it. Chapter 3 talks about the mechanical and electrical assemblies and interfaces of Ara modules. Chapter 3 also provides reference design templates for each type of module. Chapters 4 and 5 talk about power and networking between the endo and modules. Chapter 6 discusses the Ara software platform and how it interacts with various types of hardware devices. Chapter 7 talks about system-level functions, which require interactions between modules and the endo. Chapter 8 discusses environmental specifications for modules including thermal loads and electromagnetic compatibility. Chapter 9 describe the Ara module marketplace and the process for submitting and selling modules through the marketplace. Finally, Chapter 10 provides some guidelines for regulatory compliance and carrier certification.

This document provides specification requirements needed to ensure compatibility with the Ara platform. Heading titles throughout the document mark sections that contain specification language.

This document also provides reference implementation material to demonstrate exemplars that comply with the Ara specifications. Reference implementation material is in blue text throughout the document. Heading titles also mark sections that contain reference implementation material.

All MDK releases prior to release 1.0 are considered prototype releases. These are validated using prototype hardware and software implementations that frequently fall short of the objective specification. Consequently, some elements of the prototype specifications and reference implementations will differ from the objective system. Blue boxes like this one denote distinctions between the current prototype platform and the objective one throughout the document.

Contributors

The following individuals made significant contributions to the development of the MDK.

Google	Elwin Ong, Paul Eremenko, David Fishman, Jason Chua
NK Labs	Seth Newburg, Ara Knaian, Marisa Bober, Dave McCoy, Charles “Mycroft” Hannum, Boyan Kurtovich, Shahriar Kushrushahi, Nan-Wei Gong, Chris Wardman, Nate MacFadden
LeafLabs	Marti Bolivar, Perry Hung, Andrew Meyer
Rumble Development	Alan Hamilton, Mike Aronson
Toshiba	Jürgen Urban, Joerg Landmann
New Deal Design	Dan Clifton, Gadi Amit, Inbal Etgar, Susan McKinney
Motorola Mobility	Matt Mottier
Metamorph Software	Sandeep Neema, Ted Bapty

Support

The Project Ara team cannot, as a general matter, provide individualized support to module developers. If you have a comment on the MDK, have found a bug, or would like to provide feedback, you can reach the Project Ara team at ara-dev@google.com.

Developer documentation will be maintained at <http://projectara.com/mdk>, including a set of Frequently Asked Questions (FAQs) which will be updated periodically.

Developers are also encouraged to join the Ara Module Developers Google Group at <http://groups.google.com/forum/#!forum/ara-module-developers>, which serves as a forum and mailing list. If your question is not addressed in the latest documentation or FAQ, it may well have been answered in the Google Group. If not, ask away!

Limited quantities of prototype hardware, including dev boards and EPMs are available from AQS. Additionally, AQS can provide layout services for Ara form factor modules based on existing templates. You can contact AQS at ara-support@aq5-inc.com.

Revision History

Release 0.10 (alpha) - April 9, 2014 - Initial public release

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

Table of Contents

[1. Licensing](#)

[1.1. The Ara MDK License](#)

[1.2. External Licenses](#)

[1.2.1. MIPI](#)

[1.2.2. Android](#)

[1.2.3. Linux](#)

[1.2.4. Other IP](#)

[1.2.5. Prototype System External Licenses](#)

[1.2.5.1. LVDS](#)

[1.2.5.2. I2C](#)

[1.2.5.3. I2S](#)

[1.2.5.4. SDIO](#)

[1.2.5.5. DSI](#)

[2. Industrial Design](#)

[2.1. Definitions](#)

[2.2. Parceling Schemes](#)

[2.2.1. Rear Parceling Scheme](#)

[2.2.2. Front Parceling Scheme](#)

[2.3. Design Language](#)

[2.3.1. Design Language - Specification](#)

[2.3.2. Design Language - Reference Implementation](#)

[3. Mechanical and Layout](#)

[3.1. Module Geometry and Assemblies](#)

[3.1.1. Rear Module Sub-Assemblies](#)

[3.1.1.1. Rear Module Sub-Assemblies - Specification](#)

[3.1.1.2. Rear Module Sub-assemblies - Prototype Reference Implementation](#)

[3.1.1.2.1. Rear Module Templates - Prototype Reference Implementation](#)

[3.1.1.2.2. 1x2 Wi-Fi Module - Prototype Reference Implementation](#)

[3.1.2. Front Module Sub-assemblies](#)

[3.1.2.1. Front Module Sub-assemblies - Specification](#)

[3.1.2.2. Front Module Sub-assemblies - Prototype Reference Implementation](#)

[3.1.2.2.1. Media Bar Module - Prototype Reference Implementation](#)

[3.1.2.2.2. Display Module - Prototype Reference Implementation](#)

[3.1.3. Module Label](#)

[3.1.3.1. Module Label - Specification](#)

[3.1.3.2. Module Label - Reference Implementation](#)

[3.1.3.3. Module Label - Prototype Reference Implementation](#)

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

- [3.1.4. Envelope Violation Limitations](#)
 - [3.1.4.1. Envelope Violation - Specification](#)
 - [3.1.4.2. Envelope Violation - Reference Implementation](#)
 - [3.1.4.3. Envelope Violation - Prototype Reference Implementation](#)
 - [3.2. Connectors](#)
 - [3.2.1. Interface Block](#)
 - [3.2.1.1. Interface Block - Specification](#)
 - [3.2.1.2. Interface Block - Reference Implementation](#)
 - [3.2.1.3. Interface Block - Prototype Reference Implementation](#)
 - [3.2.2. Electro-permanent magnets \(EPM\)](#)
 - [3.2.2.1. Rear Module EPM - Specification](#)
 - [3.2.2.2. Rear Module EPM - Prototype Reference Implementation](#)
 - [3.2.2.3. Front Module Attachment - Specification](#)
 - [3.2.2.4. Front Module Attachment - Prototype Reference Implementation](#)
- [4. Power](#)
- [4.1. Power - Specification](#)
 - [4.1.1. Power Consumer Module - Specification](#)
 - [4.1.2. Charger Module - Specification](#)
 - [4.1.3. Power Storage Module - Specification](#)
 - [4.2. Power - Reference Implementation](#)
 - [4.2.1. LED Array Module - Prototype Reference Implementation](#)
 - [4.2.2. USB Charger Module - Prototype Reference Implementation](#)
 - [4.2.3. Battery Module - Prototype Reference Implementation](#)
- [5. Network Stack](#)
- [5.1. Module Communication Physical View](#)
 - [5.2. Network Stack](#)
 - [5.2.1. Layer 1](#)
 - [5.2.1.1. Layer 1 Capacitive Media Converter - Specification](#)
 - [5.2.1.2. Layer 1 Capacitive Media Converter - Reference Implementation](#)
 - [5.2.1.3. Layer 1 MIPI M-PHY - Specification](#)
 - [5.2.1.4. Layer 1 MIPI M-PHY - Reference Implementation](#)
 - [5.2.1.5. Layer 1 LVDS - Prototype Specification](#)
 - [5.2.1.6. Layer 1 LVDS - Prototype Reference Implementation](#)
 - [5.2.2. Layers 2, 3, 4](#)
 - [5.2.2.1. Layers 2, 3, 4 MIPI UniPro - Specification](#)
 - [5.2.2.2. Layers 2, 3, 4 UniPro - Reference Implementation](#)
 - [5.2.2.3. Layers 2, 3, 4 UniPro Tunnel - Prototype Specification](#)
 - [5.2.2.4. Layers 2, 3, 4 - Prototype Reference Implementation](#)
 - [5.2.2.4.1. I2C Tunnel - Prototype Reference Implementation](#)
 - [5.2.2.4.2. DSI Tunnel - Prototype Reference Implementation](#)

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

- [5.2.2.4.3. GPIO Tunnel - Prototype Reference Implementation](#)
 - [5.2.2.4.4. I2S Tunnel - Prototype Reference Implementation](#)
 - [5.2.2.4.5. SDIO Tunnel - Prototype Reference Implementation](#)
 - [5.2.3. Layers 5+](#)
 - [5.2.3.1. Layers 5+ - Specification](#)
 - [5.2.3.2. Layers 5+ - Prototype Reference Implementation](#)
 - [5.2.4. Network Stack Hardware - Reference Implementation](#)
 - [5.2.5. Network Stack Hardware - Prototype Reference Implementation](#)
 - [6. Software Stack](#)
 - [6.1. Kernel Drivers](#)
 - [6.1.1. Kernel Drivers - Specification](#)
 - [6.1.2. Kernel Drivers - Prototype Reference Implementation](#)
 - [6.2. Android HALs](#)
 - [6.2.1. Android HALs - Specification](#)
 - [6.2.2. HALs - Prototype Reference Implementation](#)
 - [6.3. Config Files - Prototype Reference Implementation](#)
 - [6.4. Android Application](#)
 - [6.4.1. Android Application - Specification](#)
 - [6.4.2. Android Application - Reference Implementation](#)
 - [6.4.3. Android Application - Prototype Reference Implementation](#)
 - [6.5. Prototype Development Use cases](#)
 - [6.5.1. I2C Devices](#)
 - [6.5.2. GPIO Devices](#)
 - [6.5.3. I2S Devices](#)
 - [6.5.4. SDIO Devices](#)
 - [7. System-Level Functions](#)
 - [7.1. Module Detect](#)
 - [7.2. Enumeration](#)
 - [7.3. EPM Control](#)
 - [7.4. Active Thermal Management](#)
 - [8. Environmental](#)
 - [8.1. Thermal Loads](#)
 - [8.1.1. Thermal Loads - Specification](#)
 - [8.1.2. Thermal Loads - Reference Implementation](#)
 - [8.1.3. Thermal Loads - Prototype Specification](#)
 - [8.1.4. Thermal Loads - Prototype Reference Implementation](#)
 - [8.2. Ambient Temperature and Humidity](#)
 - [8.2.1. Ambient Temperature and Humidity - Specification](#)
 - [8.2.2. Ambient Temperature and Humidity - Reference Implementation](#)
 - [8.2.3. Ambient Temperature and Humidity - Prototype Specification](#)

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

- [8.2.4. Ambient Temperature and Humidity - Prototype Reference Implementation](#)
 - [8.3. Electrostatic Discharge \(ESD\)](#)
 - [8.3.1. ESD - Specification](#)
 - [8.3.2. ESD - Reference Implementation](#)
 - [8.3.3. ESD - Prototype Reference Implementation](#)
 - [8.4. Shock](#)
 - [8.4.1. Shock - Specification](#)
 - [8.4.2. Shock - Reference Implementation](#)
 - [8.4.3. Shock - Prototype Reference Implementation](#)
 - [8.5. Vibration](#)
 - [8.5.1. Vibration - Specification](#)
 - [8.5.2. Vibration - Reference Implementation](#)
 - [8.5.3. Vibration - Prototype Reference Implementation](#)
 - [8.6. Electromagnetic Interference \(EMI\)](#)
 - [8.6.1. EMI - Specification](#)
 - [8.6.2. EMI - Reference Implementation](#)
 - [8.6.3. EMI - Prototype Reference Implementation](#)
 - [8.7. Specific Absorption Rate \(SAR\)](#)
 - [8.7.1. SAR - Specification](#)
 - [8.7.2. SAR - Reference Implementation](#)
 - [8.7.3. SAR - Prototype Reference Implementation](#)
- [9. Module Marketplace](#)
- [10. Regulatory Certification](#)
 - [10.1. US Regulations](#)
 - [10.1.1. US RF Certification](#)
 - [10.1.1.1. FCC Equipment Authorization - Specification](#)
 - [10.1.1.2. FCC Equipment Authorization - Reference Implementation](#)
 - [10.1.1.2.1. FCC Equipment Authorization - Reference Implementation for Hobbyist/Maker](#)
 - [10.1.1.2.2. FCC Equipment Authorization - Prototype/Experimental License](#)
 - [10.1.2. US Medical Device Certification](#)
 - [10.1.2.1. FDA Certification - Specification](#)
 - [10.1.2.2. Pulse Oximeter Module FDA Certification - Reference Implementation](#)
 - [10.1.2.3. FDA Certification - Prototype Reference Implementation](#)
 - [10.1.3. US Other Certification](#)
 - [10.1.3.1. US Carrier Certification](#)
 - [10.2. International Certification](#)

List of Figures

[Figure 2.1 - Ara Endo \(Medium Variant\)](#)
[Figure 2.2 - Ara Phone Built Based on Medium Endo](#)
[Figure 2.3 - Ara Rear Parceling Grid for Large, Medium, and Mini Configurations](#)
[Figure 2.4 - Valid Medium Endo Configurations \(Rear\)](#)
[Figure 2.5 - Rear Modules](#)
[Figure 2.6 - Valid Endo Configurations \(Front\)](#)
[Figure 3.1 - Rear Module Sub-assemblies](#)
[Figure 3.2 - Prototype 1x2 PCB](#)
[Figure 3.3 - Module Label Specifications](#)
[Figure 3.4 - Module Label Reference Implementations](#)
[Figure 3.5 - Pulse Oximeter Module with Y-Dimension Exceedance](#)
[Figure 3.6 - Thermal Imager Module with Z-Dimension Exceedance](#)
[Figure 3.7 - Custom Module Shell for Thermal Imager Module](#)
[Figure 3.8 - Interface Block Pinout](#)
[Figure 3.9 - Prototype Interface Block Pinout](#)
[Figure 3.10 - Rear Module EPM Exploded View](#)
[Figure 3.11 - Prototype EPM Drive Timing](#)
[Figure 3.12 - Prototype Ball-Spring Plunger Assembly](#)
[Figure 4.1 - Power Architecture](#)
[Figure 5.1 - Module to Module Communication \(with Native UniPro Support\)](#)
[Figure 5.2 - Module to Module Communication \(with UniPro Bridge ASICs\)](#)
[Figure 5.3 - Prototype Module to Module Communication](#)
[Figure 5.4 - Capacitive Media Converter](#)
[Figure 5.5 - Interface Block Unique IDs](#)
[Figure 5.6 - Network Stack over Software and Hardware](#)
[Figure 6.1 - Software Stack](#)
[Figure 6.2 - Android Configuration Files for Hardware Support](#)

List of Tables

[Table 2.1 - Ara Definitions](#)

[Table 2.2 - Design Language Geometric Guidelines](#)

[Table 3.1 - Rear Module Maximum PCB Dimensions](#)

[Table 3.2 - Prototype Rear Module PCB Available Board Area](#)

[Table 3.3 - Prototype Rear Module Z Dimension Stackup](#)

[Table 3.4 - Prototype Rear Module Templates](#)

[Table 3.5 - 1x2 Wi-Fi Module Reference Implementations](#)

[Table 3.6 - Front Modules Outer Dimensions and PCB Dimensions](#)

[Table 3.7 - Prototype Front Module Z Dimension Stackup](#)

[Table 3.8 - Prototype Media Bar Module Reference Implementation](#)

[Table 3.9 - Prototype Display Module Reference Implementation](#)

[Table 3.10 - Envelope Violation Limits](#)

[Table 3.11 - Prototype Thermal Imager Custom Shell Reference Implementation](#)

[Table 3.12 - Interface Block Pinout Descriptions](#)

[Table 3.13 - Prototype Interface Block Pinout Descriptions](#)

[Table 3.14 - Rear Module EPM Prototype Reference Implementation](#)

[Table 3.15 - Prototype Ball-Spring Plunger Assembly Reference Implementation](#)

[Table 4.1 - 1x2 Prototype LED Array Module Reference Implementation](#)

[Table 4.2 - 1x2 Prototype USB Charger Module Reference Implementation](#)

[Table 4.3. - 2x2 Prototype Battery Module Reference Implementation](#)

[Table 5.1 - Network Stack](#)

[Table 5.2 - Prototype Ara Network Stack](#)

[Table 5.3 - Toshiba UniPro Bridge ASIC Specifications](#)

[Table 8.1 - Prototype Module Thermal Loads](#)

[Table 8.2 - Prototype Ambient Temperature and Humidity Limits](#)

Acronyms

These are some of the acronyms that frequently occur in this document.

AP	Application Processor
ASIC	Application Specific Integrated Circuit
BOM	Bill of Materials
CAD	Computer-Aided Design
CMF	Color, Material, Finish
CSI	Camera Serial Interface
DSI	Display Serial Interface
EDA	Electronic Design Automation
EPM	Electro-Permanent Magnets
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input Output
HAL	Hardware Abstraction Libraries
HSIC	High-Speed Inter Chip (USB)
I2C	Inter Integrated Circuit
I2S	Integrated Interchip Sound
IP	Intellectual Property (generally not Internet Protocol)
LVDS	Low-Voltage Differential Signaling
MDK	Module Developers Kit
MIPI	Refers to mobile industry standards alliance
M-PHY	MIPI Physical Layer Specification
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PWM	Pulse Width Modulation

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

RC	Resistor-Capacitor (i.e., RC circuit)
RHC	Remote Host Controller
SDIO	Secure Digital Input Output
SLVS	Scalable Low-Voltage Signaling
STEP	Standard for the Exchange of Product Model data
UFS	Universal Flash Storage
UniPro	MIPI Unified Protocol
USB	Universal Serial Bus

1. Licensing

This section discusses certain intellectual property licensing considerations for Ara module developers. This section is a summary and not a replacement or substitute for the [MDK License Agreement](#) which governs licensing of the MDK. This summary should not be construed as legal advice.

1.1. The Ara MDK License

The MDK is licensed in accordance with the [MDK License Agreement](#). The agreement is meant to provide a permissive license to module developers, while ensuring the vibrancy and integrity of the Ara developer ecosystem in the long run. One of the more notable features of the MDK license is that it provides developers, in addition to a development license, an upfront grant of commercialization rights, conditioned only on developers “playing nice” with the rest of the Ara ecosystem. Also noteworthy is that the MDK license does not alter any of the standard open source software licenses associated with Android or any Ara-specific drivers, apps, etc. that are distributed as part of the MDK. The MDK license does not allow--nor does the MDK specification facilitate--the development or commercialization of Ara endoskeletons by third-party developers.

1.2. External Licenses

Project Ara strives to embrace industry standards where possible. Consequently, the MDK references numerous external standards and specifications. These are available in accordance with their specific license agreements and are neither owned by Google nor part of the MDK. Below is a brief summary of these external licenses.

1.2.1. MIPI

The MIPI Alliance is an open membership industry consortium that developers interface specifications for mobile devices. Ara modules implement several MIPI specifications. At minimum, modules communicate with the endo and with other modules using the MIPI M-PHY and UniPro specifications. Depending on the application, some modules may implement additional MIPI specifications such as CSI and DSI. In order to get a hold of the detailed MIPI specification, you need to join the MIPI Alliance. This gives you a royalty-free license to implement your own version of the MIPI interfaces. Alternatively, developers may procure from third-party vendors IP blocks (for use in FPGAs or in creating ASICs) or complete bridge chips implementing the MIPI standard interfaces.

1.2.2. Android

The software stack for the Ara platform is built on the open source Android OS. The majority of [Android](#) software including Android software modified/developed for Ara is provided under the [Apache 2.0 license](#). Except for the preservation of the copyright notice and disclaimer, this license allows the user of the software the freedom to use the software for any purpose, to

distribute it, to modify it, and to distribute modified versions of the software, under the terms of the license, without concern for royalties.

1.2.3. Linux

The Android OS is built on the Linux kernel. The Linux kernel and its derived works, including kernel drivers, are distributed under the terms of the [GNU General Public License, version 2](#).

1.2.4. Other IP

To the extent that you utilize intellectual property from other sources, you are responsible for obtaining approved licenses to implement and market their modules.

1.2.5. Prototype System External Licenses

1.2.5.1. LVDS

The current prototype platform uses LVDS at the physical network layer (OSI Layer 1). LVDS is specified and copyrighted by the Telecommunications Industry Association (TIA) under the official technical standard TIA/EIA-644-A. The prototype system uses a Lattice Semiconductors Field-Programmable Gate Array (FPGA) that implements LVDS and meets all necessary copyright and licensing requirements. A copy of the LVDS standards document may be purchased from [TIA](#).

1.2.5.2. I2C

The current prototype platform uses the I2C bus standard implemented as a tunneled protocol between modules and the Application Processor. The I2C bus standard is copyrighted by NXP Semiconductors, and is provided free of charge on their [website](#).

1.2.5.3. I2S

The current prototype platform uses the I2S bus standard implemented as a tunneled protocol between modules and the Application Processor. The I2S standard is available online [here](#) and [here](#).

1.2.5.4. SDIO

The current prototype platform uses the SDIO bus standard implemented as a tunneled protocol between modules and the Application Processor. The standard is copyrighted by the [SD Card Association](#) and provided free on their [website](#).

1.2.5.5. DSI

In the current prototype platform, the DSI standard is implemented as a tunneled protocol between the prototype Display Module and the Application Processor. This standard is copyrighted by the [MIPI Alliance](#) and require an approved license to implement.

2. Industrial Design

2.1. Definitions

The Ara architecture requires the introduction of several new concepts to the traditional mobile device lexicon. These terms are defined in Table 2.1 below and illustrated in Figures 2.1 and 2.2.

Table 2.1 - Ara Definitions

Term	Definition
Module	Modules are the building blocks of an Ara phone. They are the hardware analogue to software apps. These are physical components that implement various phone functions. There are currently two major classes of modules: Front modules, which make up the front of the phone and generally provide user interaction or interface affordance such as the display, speaker, microphone, etc., and rear modules, which provide the bulk of the phone's back-end (non-user facing) functionality. Front modules reach across the entire width of a particular endoskeleton frame, while rear modules come in three standard sizes (1x1, 1x2, and 2x2) and can fit into multiple frame sizes. (See Figure 2.3)
Endoskeleton (Endo)	The Ara endoskeleton (or "endo") is the frame and backplane of the device, determining the size and layout of the phone. Ara modules slide in and attach to the endo's slots, which has a backplane to electrically and logically connect modules together. There are currently three endo size variants: Mini, Medium, and Large, with varying rib configurations for each. Note that the Large endo variant and respective modules will be defined in a future MDK release. Note also that the MDK only details the specification of the endo to the extent that it is necessary for module developers to develop modules. In the interest of maintaining the integrity of the Ara platform specification, third-party endo development is not permitted.
Spine (Endo Spine)	A singular vertical feature that bisects the rear of the endo and forms part of the module slots. (See Figures 2.1 and 2.2)
Rib (Endo Rib)	Horizontal features located either in the front or the rear of the endo and forms part of the module slots. Note that the rib configuration shown in Figures 2.1 and 2.2 is an instance of a rib arrangement and not the only possible arrangement. (See Figure 2.4)
Top	The orientation of the primary display module determines the "top" of the phone. The top of the phone is defined as the direction in which the volume buttons affixed to the display module are biased. (See Figure 2.2)

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

Phone Coordinate Axes (X, Y, Z)	The Ara platform uses the Android defined coordinate system . The Side-to-Side direction defines the X axis. The Top-Bottom direction defines the Y axis. The thickness direction defines the Z axis. (See Figure 2.2)
Interface Block	The interface block is the area on the endo and the modules where the electrical power pins and capacitive data pads are located.
Electro-Permanent Magnets (EPM)	Rear modules attach and secure themselves to the endo with electro-permanent magnets (EPM) directly, whereas front modules utilize EPM-actuated latches or pins for attachment to the endo.
Module Shell	The module shell is a user-replaceable cover for Ara modules that can be aesthetically customized and is 3D printed as part of the Ara fulfillment process. With a few exceptions as noted in the MDK, Ara modules should nominally support user serviceable module shells.



Figure 2.1 - Ara Endo (Medium Variant)

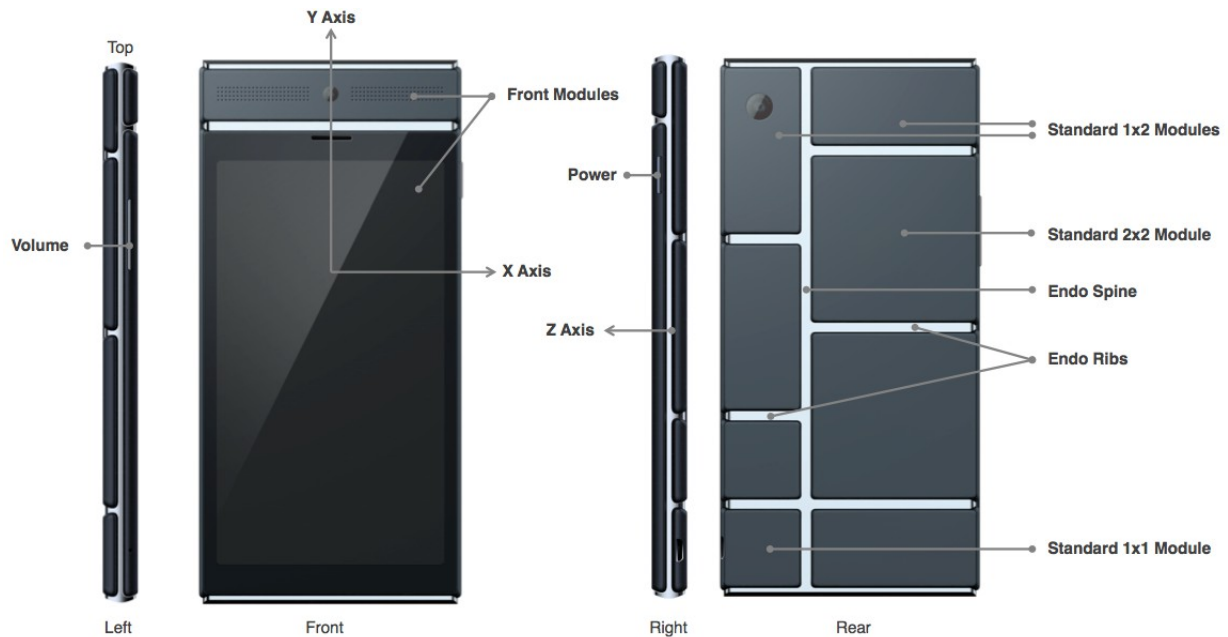


Figure 2.2 - Ara Phone Built Based on Medium Endo

2.2. Parceling Schemes

Parceling schemes are rules that determine how and where modules can be placed in the endo frame. The front parceling scheme only has a few possible layouts, while the rear parceling scheme can have many different configurations depending on the location of the ribs and spine.

2.2.1. Rear Parceling Scheme

The rear of the endo is parceled into 1x1 unit squares. Each 1x1 square is approximately 20 mm. Note, however, that 1x2 and 2x2 modules are not exactly 20x40mm and 40x40mm due to the fact that the thickness of the missing rib must be accounted for to conform to the overall grid scheme. Refer to the computer-aided design (CAD) models and drawings for exact dimensions. All three endo sizes use the same rear parceling scheme.

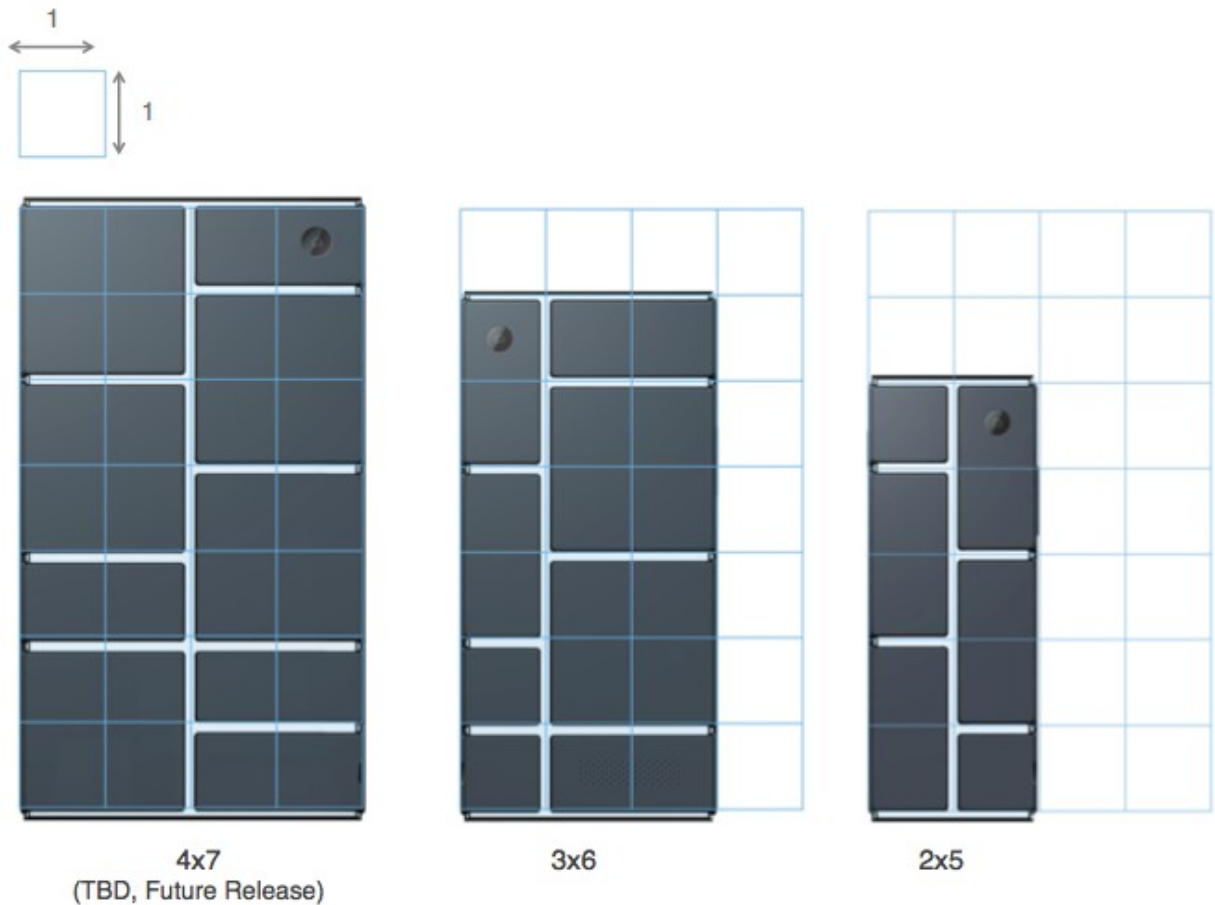


Figure 2.3 - Ara Rear Parceling Grid for Large, Medium, and Mini Configurations

As shown in Figure 2.3, a Medium endo variant is composed of a 3x6 parceled grid while the Mini and Large variants are 2x5 and 4x7 (TBD) respectively. Furthermore, the following design rules govern the placement of the spine and ribs on the rear of the endo:

- Endos must have exactly one vertical spine.
- The Medium variant spine must be at a 1:2 horizontal offset from the centerline of the device as viewed from the rear.
- The Mini and Large (TBD) variant spine must be in the middle.
- For the horizontal ribs, there must be at least 1 rib per 2 units (since modules cannot be larger than 2x2).
- Only a single cross is allowed in an endo (that is, only one rib can go straight across the spine on both sides).

The application of these design rules results in a discrete set of possible endo configurations for each size variant. Figure 2.4 shows the valid set of rear endo configurations for the Medium

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

variant. While Google may or may not make every possible endo variant available, developers should be mindful that modules should support every variant.

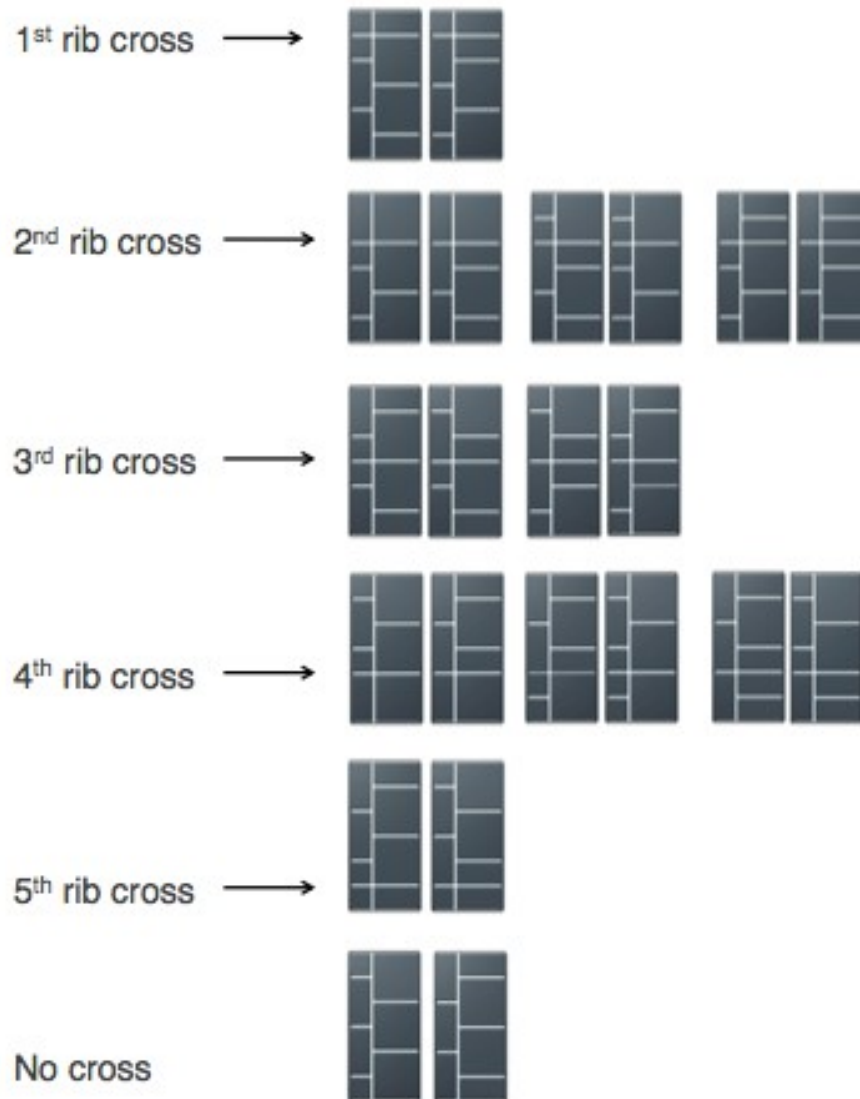


Figure 2.4 - Valid Medium Endo Configurations (Rear)

The Ara parceling scheme and endo design rules enable the three rear modules to be used across multiple endo sizes. A 1x2 module can be used on all three endo size variants, while a 2x2 module can be used on the Large and Medium endo variants, and a 1x1 module can be used on the Medium and Mini endo variants. Note also that a 1x2 module can be inserted into an endo either in the vertical or horizontal orientation.

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

Figure 2.5 shows the front and rear views of rear modules with standard module shells installed. Note that all rear modules should nominally support user-replaceable module shells. Module shells attach to rear modules via mechanical snap-fit features built into the module base and shell itself.

Replaceable module shells are a unique feature of the Ara architecture. They allow users to leverage consumer-grade, full-color 3D printing to aesthetically customize their Ara phone before purchase, and if desired, to replace each module shell any time thereafter.

Figure 2.5 also shows the locations of the interface blocks for each module. Note that 2x2 modules may support an optional second interfaces block for increased power and data utilization.

In the current prototype platform, due to the use of FPGAs for hosting the network stack, it is presently infeasible to configure a 1x1 module. Furthermore, due to limitation on the number of network switch ports, modules can only support a single interface block per module, i.e., 2x2 modules do NOT support the second optional interface block.

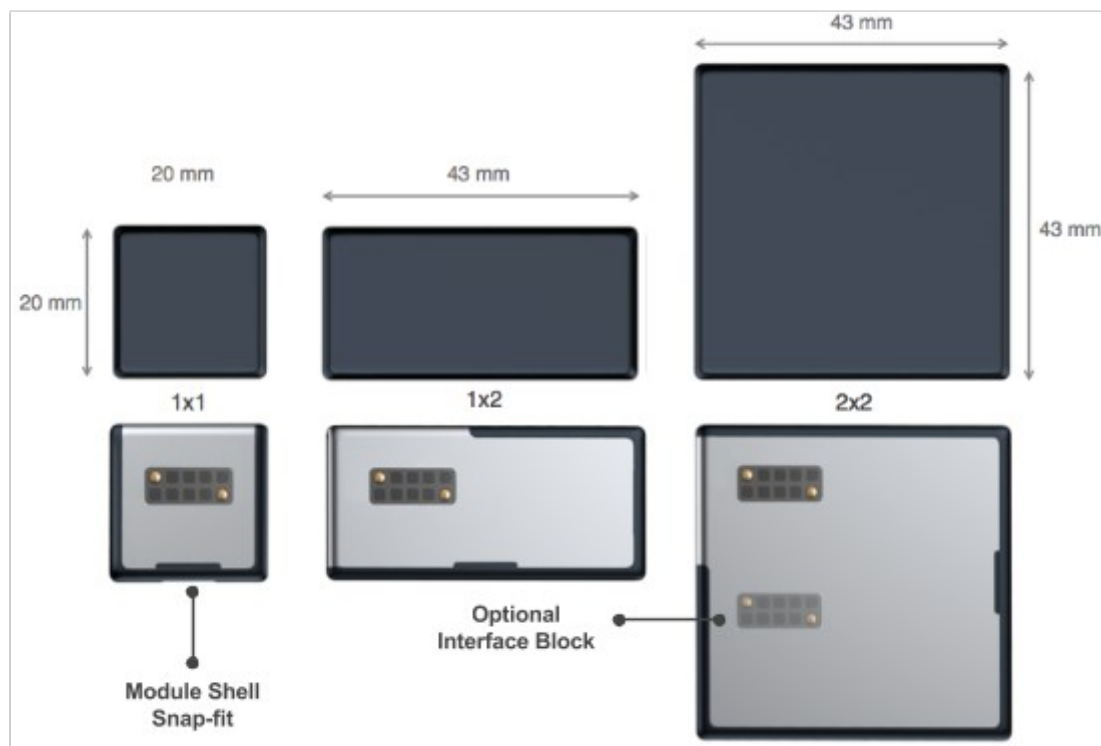


Figure 2.5 - Rear Modules

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

2.2.2. Front Parceling Scheme

The following design rules govern the front parceling scheme:

- Vertical spines are not allowed - all modules must fill the complete horizontal width
- A maximum of 2 ribs are allowed
- Only a single rib is allowed in each of the upper or lower halves

The parceling scheme for the front endo results in modules that are proportionally sized for each endo size variant. Figure 2.6 shows the valid set of front endo configurations for each size variant, labeled A through L. Front modules for the Large endo variant will be formalized in a future MDK release.

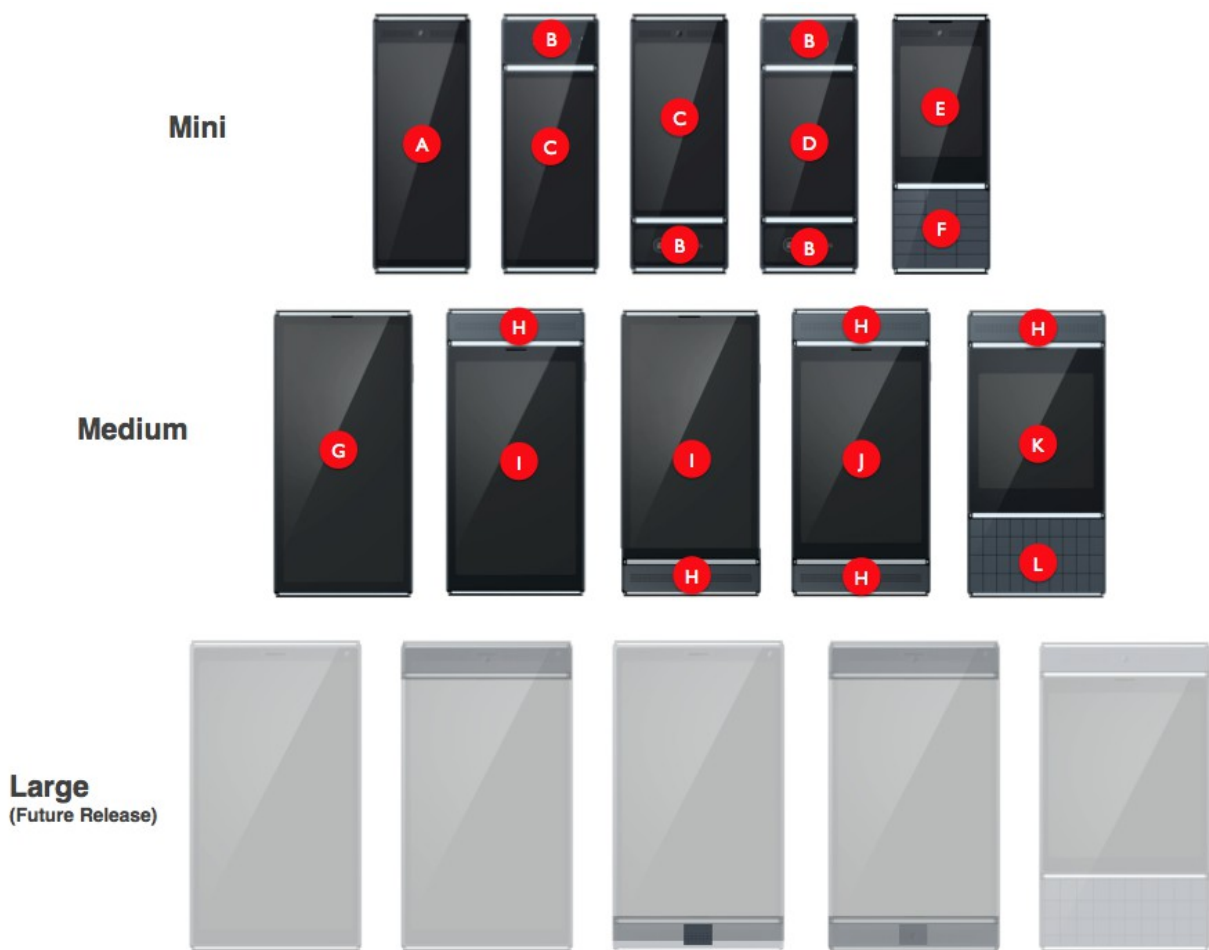


Figure 2.6 - Valid Endo Configurations (Front)

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

2.3. Design Language

The design language is a set of design elements used to visually communicate a specific aesthetic. Implementing a consistent design language ensures that every Ara phone has a set of aesthetically cohesive modules, even though each phone may include modules from different sources.

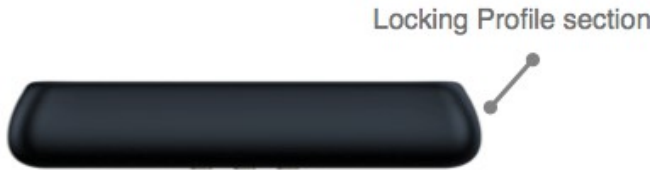

2.3.1. Design Language - Specification

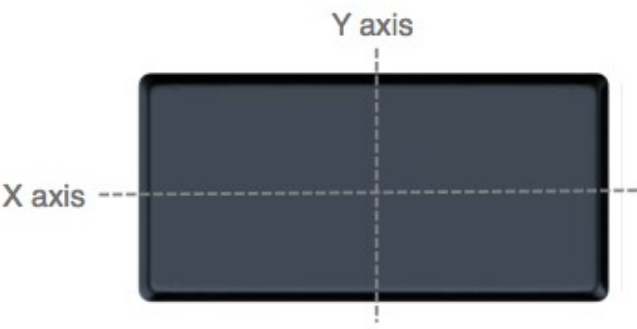

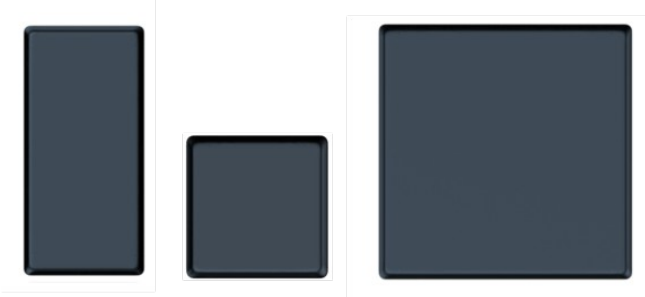
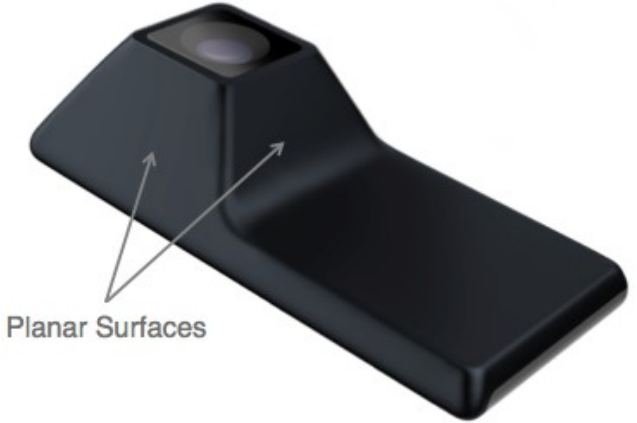
The overall goal of the module aesthetic is to create a smooth, flat, pebble form. The reasons for this aesthetic are as follows:

1. A softer form without sharp lines and edges is simple, iconic, and visually easy to understand
2. The shape enables modules to easily slide into a module slot
3. The form of the module is one that is friendly to hold and enjoyable to handle

The design language can be articulated as a set of geometric and color, material, finish (CMF) guidelines. Table 2.2 summarizes the geometric guidelines. CMF guidelines will be provided in a future release.

Table 2.2 - Design Language Geometric Guidelines

Guideline	Illustration
Module side profile must conform to the shape of the endo. This is not only necessary to follow the design language, but is also critical to allow the modules to slide into module slots in the endo and lock into place.	 A side view of a dark, rounded rectangular module. A line points to the top edge, labeled "Locking Profile section".
Modules taller in Z should continue the trapezoid form while expanding in Z.	 A side view of a dark, rounded rectangular module. An upward-pointing arrow is labeled "Z axis".

<p>The side profile sections should be symmetric across X and Y axes.</p>	
<p>Corners should have 1.5 mm curvature radii.</p>	
<p>Modules should keep rectilinear footprints when possible; these are exemplified by 1.5 mm rounded corners connected by straight lines.</p>	
<p>Planar surfaces (meeting at angles) do not need to be parallel or perpendicular to the module's bottom surface.</p>	

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

2.3.2. Design Language - Reference Implementation

Section 3 provides example applications of the Ara design language including references to CAD models for module templates and custom enclosures that meet the Ara design language specifications.

3. Mechanical and Layout

This section defines mechanical specifications and general layout for Ara modules.

3.1. Module Geometry and Assemblies

3.1.1. Rear Module Sub-Assemblies

3.1.1.1. Rear Module Sub-Assemblies - Specification

All three rear module types are composed of three sub-assemblies: the module base, printed circuit board (PCB), and safety shield. As described in Section 2, the module shell is not considered part of the module assembly. To the extent that it may deviate from a standard shell geometry, it will be created in accordance with a module developer's geometric specifications, but will include a consumer's aesthetic customizations and therefore will be manufactured as part of the fulfillment process (i.e., not by the module developer). Figure 3.1 shows an exploded view of these sub-assemblies for a 1x2 module.

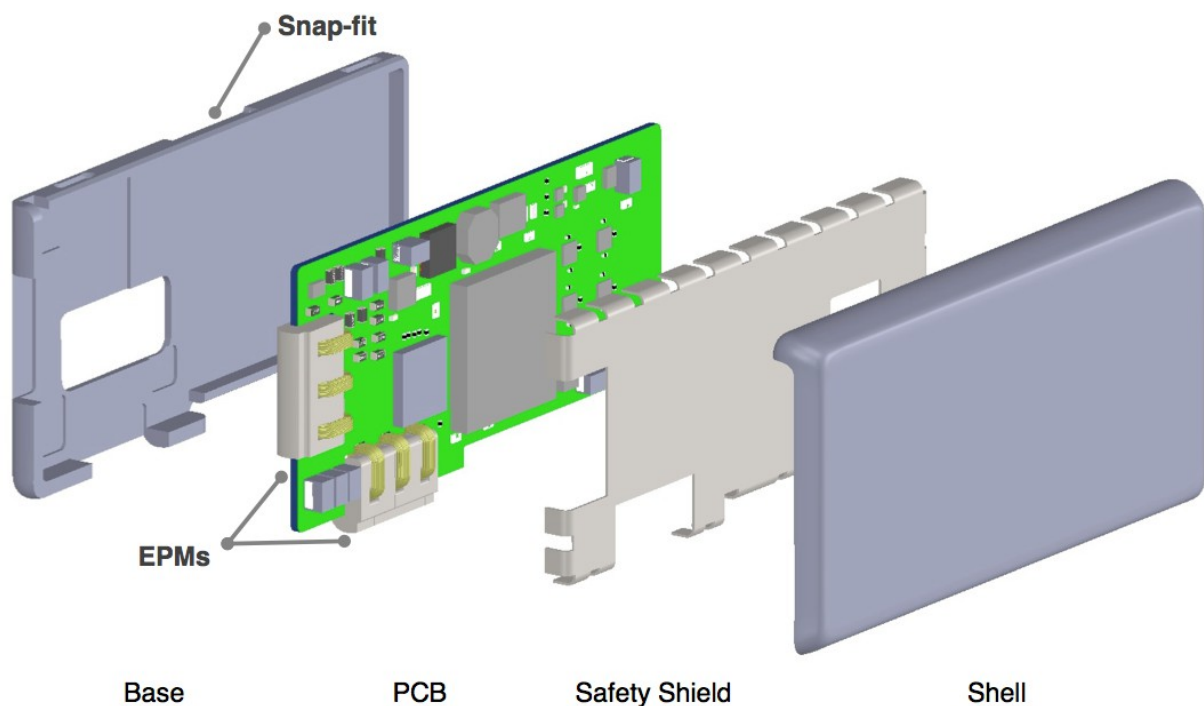


Figure 3.1 - Rear Module Sub-assemblies

The module base must be a single piece of machined 6061 aluminum. Except for cutouts needed for external interfaces (e.g., USB connector), the external dimensions of the module base must conform to the shape defined by the CAD model and drawings provided in the

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

reference implementation. This requirement ensures the module is compatible with the 3D-printed shells and fits snugly into the endo frame. At least the external face of the module base must also be anodized with a satin finish (MT110005) to provide a consistent look and feel.

The PCB contains the circuitry for the module, including the endo interface functionality, as well as the custom functionality of the module. Any non-electronic components that are part of the module (e.g., batteries, sensors) must either mount to or in place of the PCB. Table 3.1 provides maximum dimensions available for rear module PCBs. PCBs smaller than these dimensions are allowed. Note that interface block(s) and EPM(s) (and driver circuits) will take up some of the available PCB areas. The PCB must mount to the module base.

Table 3.1 - Rear Module Maximum PCB Dimensions

Rear Module	PCB Maximum Dimensions (X, Y)
1x1	18 mm x 18 mm (TBD)
1x2	18 mm x 40.5 mm (TBD)
2x2	39.5 mm x 41 mm (TBD)

Table 3.3 provides the approximate areas available in the prototype module PCBs for custom module circuitry. The PCB contains prototype EPM driver circuits and an FPGA to handle inter-module communications. Figure 3.2 shows a rendering of the 1x2 module template PCB, which includes the set of circuits and components needed by the prototype. Due to the PCB area taken up by these prototype circuits, there is insufficient space to support custom module circuitry in the 1x1 module. More PCB area is expected in the objective system as functions in the FPGA and EPM driver circuits are migrated to custom ASICs with smaller footprints.

Table 3.2 - Prototype Rear Module PCB Available Board Area

Rear Module	Approximate PCB Area
1x2	600 mm ²
2x2	1490 mm ²

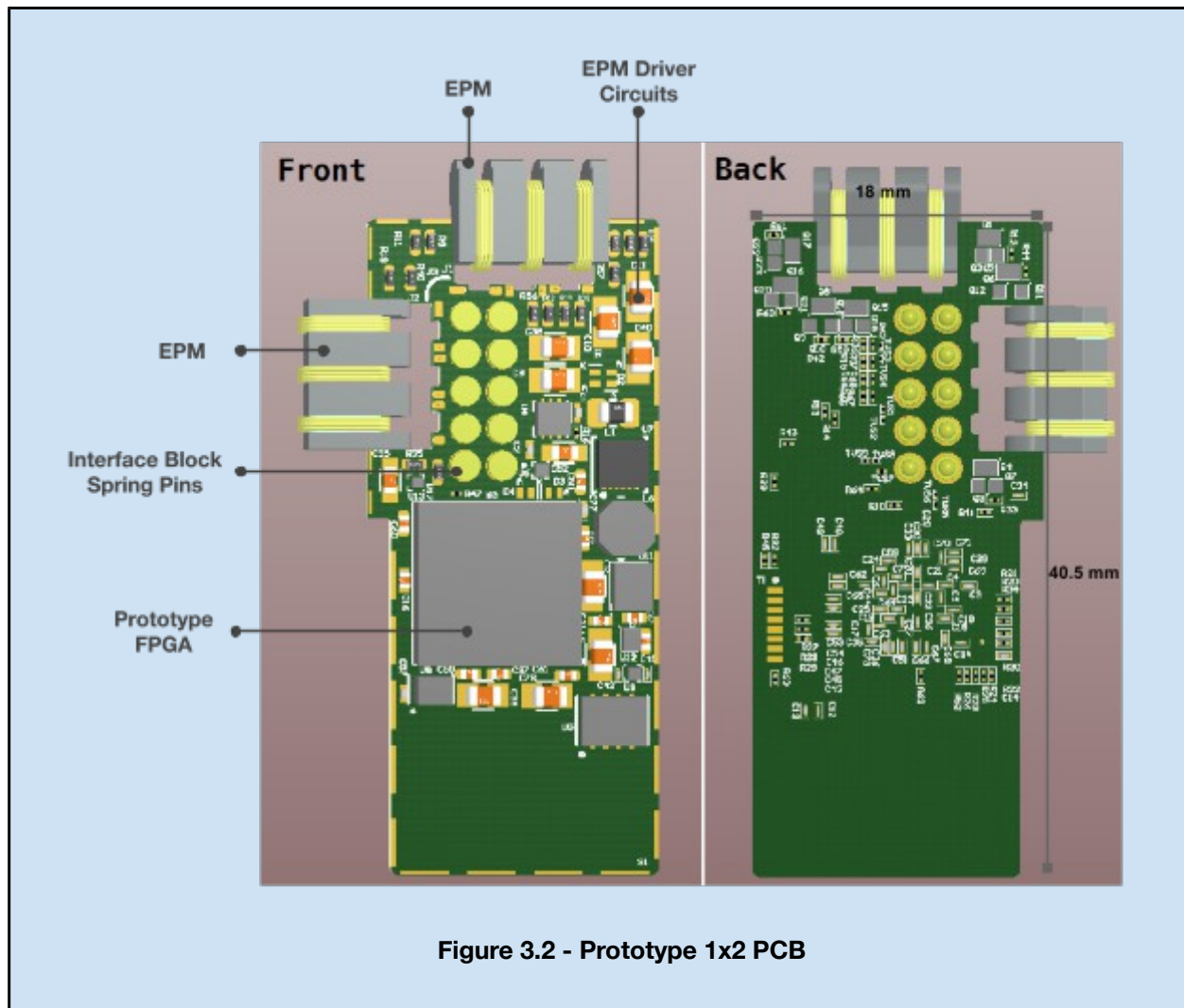


Figure 3.2 - Prototype 1x2 PCB

A safety shield is necessary to prevent users from making unintentional contact with sensitive components on the PCB while changing module shells. The safety shield also acts as a Faraday cage to protect modules from potential interference and ensure a uniform RF environment for modules which are intentional RF emitters. The safety shield must be made of nickel-plated steel or a functionally and aesthetically similar metal and securely mount to the module base.

3.1.1.2. Rear Module Sub-assemblies - Prototype Reference Implementation
Table 3.2 provides a vertical stack up of various layers in prototype rear modules including dimensions available for PCB components.

Table 3.3 - Prototype Rear Module Z Dimension Stack Up

Layer	Thickness	Notes:
Module Shell	0.65 mm	
Safety Shield	0.26 mm	Includes clearance above safety shield; safety shield is 0.18 mm thick
Clearance	1.65 mm	Available for PCB component
PCB	0.56 mm	FR-4 PCB
Clearance	0.36 mm	Available for PCB component. Some areas in the prototype may have an additional 0.22 mm due to cutouts in the base (e.g. for EPM driver circuit). See CAD for details.
Insulation	0.02 mm	E.g. Kapton tape
Module Base	0.5 mm	
Total Module Thickness	4.0 mm	

The following subsections provide reference implementations of several prototype rear modules.

3.1.1.2.1. Rear Module Templates - Prototype Reference Implementation
Table 3.4 provides relative file paths to design artifacts to create a minimal instantiation of prototype rear modules. The module templates include both mechanical and electrical models. The mechanical model includes 3D CAD models in STEP format and additional CAD line drawings. Electronic design automation (EDA) files are in Altium format. The template also includes EDA output packages, which contain schematics, bill-of-materials (BOM), and PCB layout guidelines.

The module template EDA and output files include the prototype EPM driver circuit and FPGA that handles module communications and other system-level functions. The FPGA provides tunneling of several interface protocols that module developers can use to communicate with the TI OMAP 4460 Application Processor (AP) in the AP module or on the AP development board. The Network Stack section of this document details these protocols.

Table 3.4 - Prototype Rear Module Templates

Prototype Rear Module	Design Artifacts
Prototype 1x2 Module	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\1x2\Prototype1x2ModuleTemplate\CADFiles EDA File and Outputs: ReferenceMaterials\1x2\Prototype1x2ModuleTemplate\EDAFiles
Prototype 2x2 Module	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\2x2\Prototype1x2ModuleTemplate\CADFiles EDA File and Outputs: ReferenceMaterials\2x2\Prototype1x2ModuleTemplate\EDAFiles

3.1.1.2.2. 1x2 Wi-Fi Module - Prototype Reference Implementation

Table 3.5 provides relative file paths to design artifacts to create the prototype Wi-Fi module in a 1x2 module template.

Table 3.5 - 1x2 Wi-Fi Module Reference Implementations

Module	Design Artifacts
Prototype Wi-Fi Module	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\1x2\PrototypeWi-FiModule\CADFiles EDA File and Outputs: ReferenceMaterials\1x2\PrototypeWi-FiModule\EDAFiles

3.1.2. Front Module Sub-assemblies

3.1.2.1. Front Module Sub-assemblies - Specification

Figure 2.6 shows the possible distinct front module (A-L). While all front modules require a module base and PCB for mounting the interface block(s) and EPDs, the other sub-assemblies are specific to the module. As an example, a Display Module may have a display, touch sensor, and glass cover. Table 3.6 provides outer dimensions and dimensions available for a PCB in the two front modules in the current prototype. Maximum PCB dimensions for the other modules will be provided in a future MDK release.

Table 3.6 - Front Module Outer Dimensions and PCB Dimensions

Front Module (Endo Size)	Module Outer Dimensions (X, Y)	Maximum PCB Dimensions (X, Y)
A (Mini)	45 mm x 114 mm	TBD
B (Mini)	45 mm x 20 mm	TBD
C (Mini)	45 mm x 91 mm	TBD
D (Mini)	45 mm x 68 mm	TBD
E (Mini)	45 mm x 68 mm	TBD
F (Mini)	45 mm x 46 mm	TBD
G (Medium)	67.82 mm x 136.87 mm	TBD
H (Medium)	67.82 mm x 14.82 mm	62.8 mm x 9.5 mm
I (Medium)	67.82 mm x 121.02 mm	62.5 mm x 115.7 mm
J (Medium)	67.82 mm x 104.25 mm	TBD
K (Medium)	67.82 mm x 75.02 mm	TBD
L (Medium)	67.82 mm x 44.81 mm	TBD

3.1.2.2. Front Module Sub-assemblies - Prototype Reference Implementation

Table 3.7 provides a vertical stack up of various layers in a front module including dimensions available for PCB components, displays, glass, and other components that are typically found on the front of a phone.

Table 3.7 - Prototype Front Module Z Dimension Stack Up

Layer	Thickness	Notes:
Module Shell	1.0 mm	
Clearance	1.69 mm	Available for module component
PCB	0.56 mm	FR-4 PCB
Clearance	0.23 mm	Available for module component
Insulation	0.02 mm	E.g. Kapton tape
Module Base	0.5 mm	
Total Module Thickness	4.0 mm	

3.1.2.2.1. Media Bar Module - Prototype Reference Implementation

The current prototype implementation includes two front modules, both for the Medium endo variant. There is a Media Bar Module (Class H) and a Display Module (Class I). Table 3.8 provides relative file paths to design artifacts to create a prototype Media Bar Module.

Table 3.8 - Prototype Media Bar Module Reference Implementation

Component	Design Artifacts
Media Bar Module	<ul style="list-style-type: none">3D Models and Drawings: ReferenceMaterials\ClassH(Medium)\PrototypeMediaBarModule\CADFilesEDA File and Outputs: ReferenceMaterials\ClassH(Medium)\PrototypeMediaBarModule\EDAFiles

3.1.2.2.2. Display Module - Prototype Reference Implementation

Table 3.9 provides relative file paths to design artifacts to create a prototype Display Module.

Table 3.9 - Prototype Display Module Reference Implementation	
Component	Design Artifacts
Display Module	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\Classl(Medium)\PrototypeDisplayModule\CADFiles EDA File and Outputs: ReferenceMaterials\Classl(Medium)\PrototypeDisplayModule\EDAFiles

3.1.3. Module Label

3.1.3.1. Module Label - Specification

The back of every module must have a clearly marked module label. This label must include the Ara emblem, the name of the module developer, the name of the module, and a module icon signifying the module function. The Ara emblem, module icons, and application instructions will be provided in a future MDK release. If applicable, module labels should include regulatory markings such as the module FCC ID and CE markings.

Figure 3.3 illustrates the rules for positioning text and icons in the module label in relation to the interface block. If a module has multiple interface blocks, the module label must be positioned in relation to the top most interface block. Positioning of the module identification must fall 2 mm under the interface block, center aligned with the interface block. Module identification text must use 4 pt Helvetica Condensed font. Positioning of the regulatory markings must be 1 mm above the interface block. Regulatory identification text must be 3.5 pt Helvetica Condensed font.

Module labels should be applied with a laser-etched process.

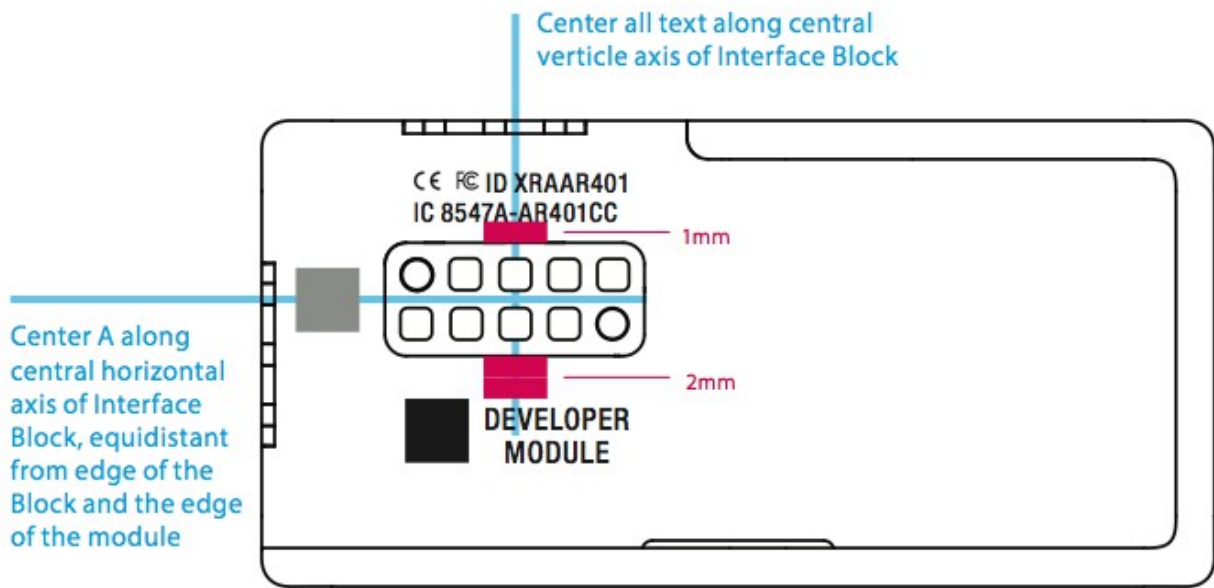


Figure 3.3 - Module Label Specifications

3.1.3.2. Module Label - Reference Implementation

Figure 3.4 illustrates compliant module labels. While only rear modules are shown, the specifications similarly apply to front modules, in relation to their top-most interface block. These reference implementations will be updated with icons in a future release.

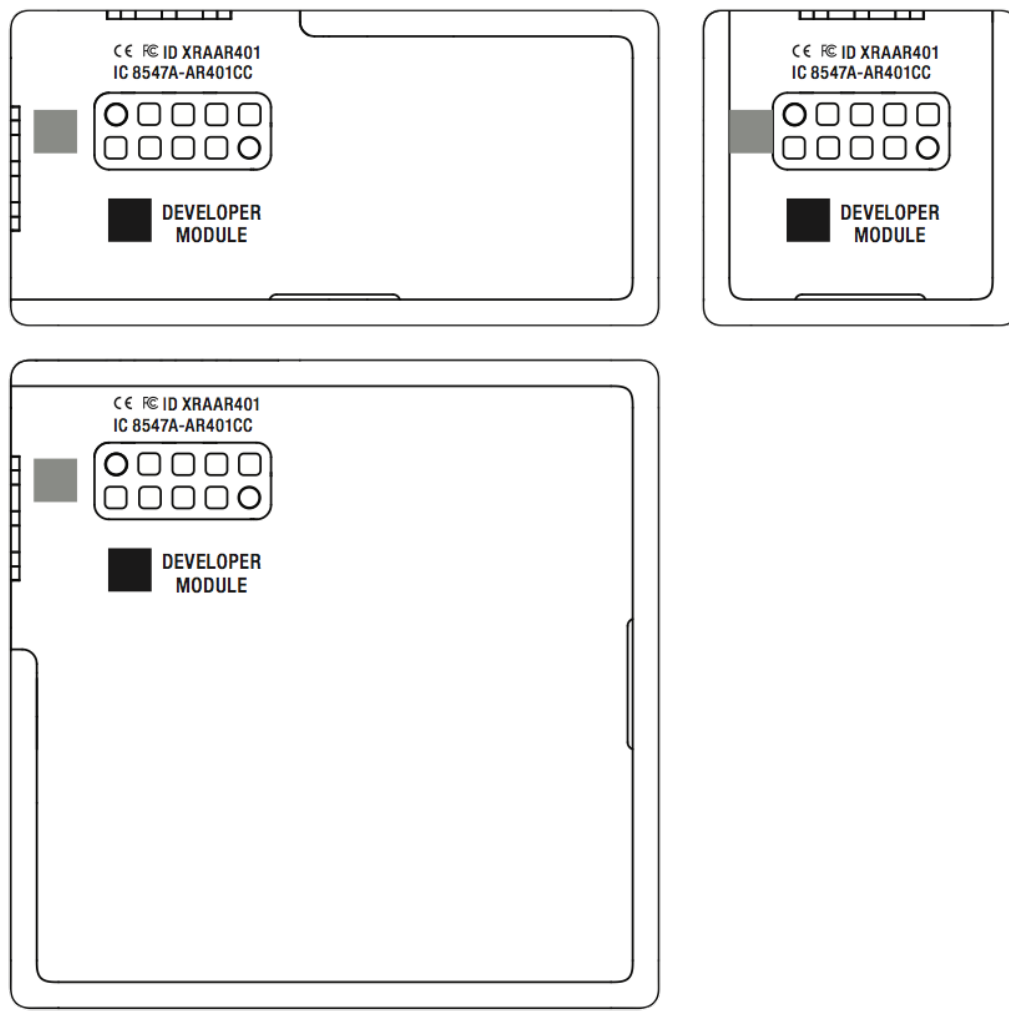


Figure 3.4 - Module Label Reference Implementations

3.1.3.3. Module Label - Prototype Reference Implementation
 Module labels are not required for prototypes.

3.1.4. Envelope Violation Limitations

3.1.4.1. Envelope Violation - Specification

The envelope for a module comprises the standard dimensions for the module to conform to the Ara endos. While a module will ideally fit within these very specific dimensions, modules are allowed to exceed the standard envelope in the Y (top-bottom) and Z (thickness) directions. Modules are NOT allowed to exceed the envelope in the X (side-side) direction. Table 3.10 provides the absolute maximum exceedance limits, driven by the capability of 3D printers (to

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

print the associated module shells); however, module developers should also use good engineering judgement to determine the stress levels from user levied forces and torques and ensure exceedances are structured to accommodate common usage scenarios including scenarios where the module and phone are in the user's pocket. Future releases of the MDK may be more restrictive in exceedance limits based on results of handling and accelerated lifecycle testing. Modules with dimensional exceedances must have a custom safety shield and ensure all electrically active components are encapsulated within.

Table 3.10 - Envelope Violation Limits

Direction	Max Dimension	Notes
X	45 mm (Mini), 67.02 mm (Medium), 21.82 mm (1x1, 1x2), 44.82 mm (1x2, 2x2)	Modules are NOT allowed to exceed the standard envelope in the X direction.
Y	20 cm	Overall module height must not exceed this figure.
Z	2.5 cm	Overall module thickness must not exceed this figure.

Whenever modules exceed the standard envelope, the exceedance should conform to the Ara design language specification (in Section 2).

3.1.4.2. Envelope Violation - Reference Implementation

Exceedances may be appropriate and necessary for some applications as demonstrated in Figures 3.5 and 3.6. Figure 3.5 is a reflective Pulse Oximeter Module, which measures blood oxygen saturation. The exceedance in the Y dimension provides a convenient affordance and sensor location for a user's finger.

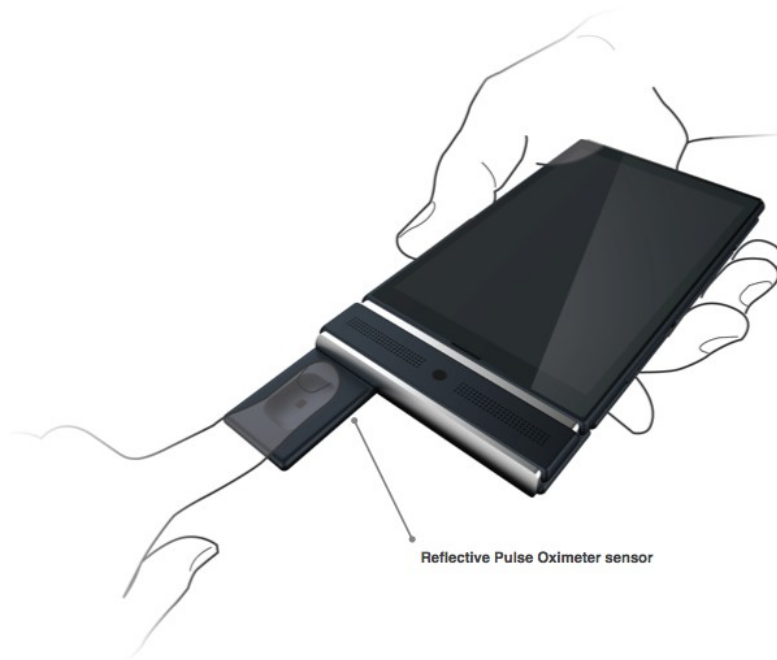


Figure 3.4 - Pulse Oximeter Module with Y-Dimension Exceedance

3.1.4.3. Envelope Violation - Prototype Reference Implementation

Figure 3.6 is an example of a Prototype Thermal Imager Module. The exceedance in the Z dimension enables modules to accommodate components with high thickness requirements such as a camera lens unit.



Figure 3.6 -Thermal Imager Module with Z-Dimension Exceedance

Figure 3.7 demonstrates a compliant custom module shell for the prototype Thermal Imager Module. The shell has a raised opening for the lens. Table 3.11 provides relative paths to design artifacts for the custom shell.

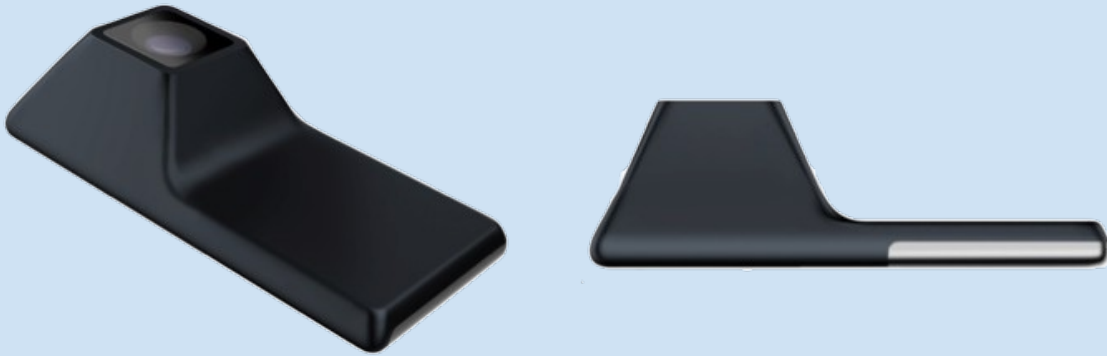


Figure 3.7 - Custom Module Shell for Thermal Imager Module

Table 3.11 - Prototype Thermal Imager Custom Shell Reference Implementation

Component	Design Artifacts
Prototype Thermal Imager Custom Shell	<ul style="list-style-type: none"> 3D Model STEP File: ReferenceMaterials\1x2\PrototypeThermalImagerModule\CADFiles\PrototypeThermalImagerModuleShellModelAlpha.STEP CAD Drawing: ReferenceMaterials\1x2\PrototypeThermalImagerModule\CADFiles\PrototypeThermalImagerModuleShellDwgAlpha.pdf

3.2. Connectors

The following subsections define the two connectors that are required in all Ara modules: Interface blocks (for data and power to exchange between modules and the endo) and EPMs (to keep the modules mechanically attached in the X direction). A standard antenna connector will be specified for modules that require antennas in a future MDK release.

3.2.1. Interface Block

The interface block hosts electrical and data connectors for the module-to-endo interface. The Ara platform utilizes a contactless capacitive interface for high-speed data transfer between modules and the endo. The Network Stack section details the capacitive data transfer mechanism.

3.2.1.1. Interface Block - Specification

Each interface block is composed of 2 power pins and 8 capacitive data pads. The interface block is itself mounted and printed on the module PCB. The module templates EDA files detail geometric specifications of an interface block.

Figure 3.8 and Table 3.12 detail the “pinouts” of the interface block. The 8 data pads correspond to 4 pairs of differential data lines and 2 bi-directional data lanes (A and B).

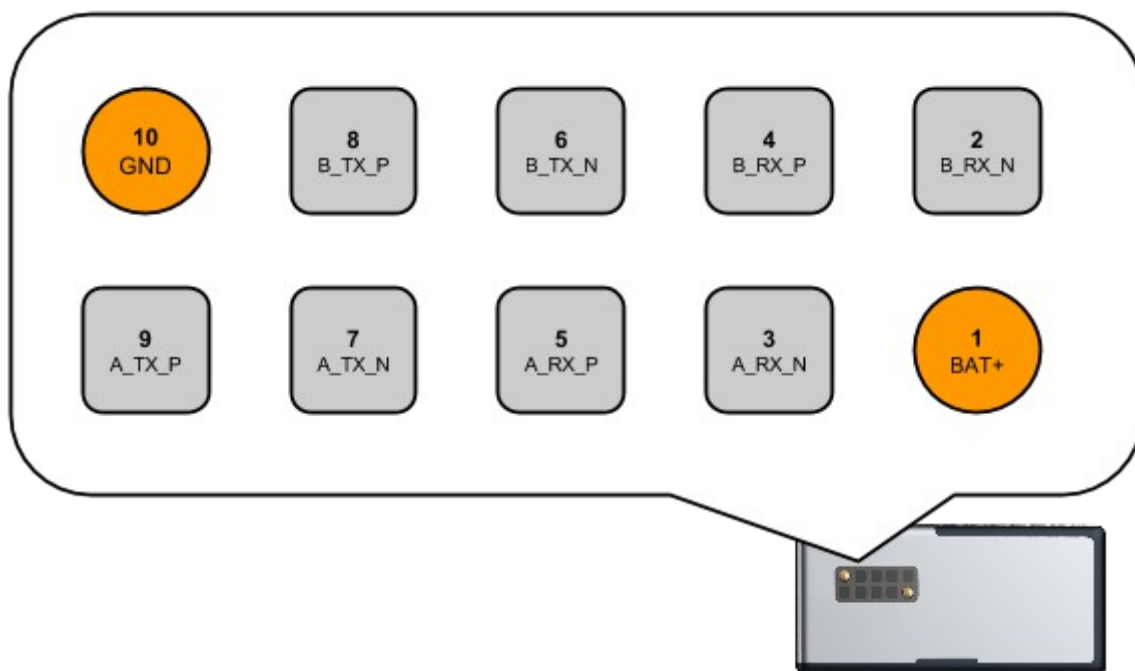


Figure 3.8 - Interface Block Pinout

Table 3.12 - Interface Block Descriptions

#	Pin/Pad Label	Description
1	+BATT	Power
2	B_RX_N	B Receive Negative
3	A_RX_N	A Receive Negative
4	B_RX_P	B Receive Positive
5	A_RX_P	A Receive Power
6	B_TX_N	B Transmit Negative
7	A_TX_N	A Transmit Negative
8	B_TX_P	B Transmit Positive
9	A_TX_P	A Transmit Positive
10	GND	Ground

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

The power pins (+BATT and GND) must be spring loaded and support a current rating of at least 1A. The spring pins must fit within a 2.8 mm square.

Data pads must be made from ½ ounce copper on 0.5 mm FR4 2-layer PCB. Each data pad is approximately 2.8 mm square (module template EDA files detail the exact shape and dimensions). Vias on the data pads must be filled and tented; black solder mask must be applied to cover the whole interface block.

3.2.1.2. Interface Block - Reference Implementation

An example compliant power pin is the [MillMax 0965-0-15-20-80-14-11-0 Spring Pin](#). Example compliant data pads will be provided in a future MDK release.

3.2.1.3. Interface Block - Prototype Reference Implementation

The interface block in the current prototype platform differs from the objective specification. Instead of contactless data transfer with capacitive pads, the prototype uses electrical signaling with LVDS and spring pins for data transfer in addition to power. The module template section provides design artifacts for the prototype interface block reference implementation.

The prototype interface block uses 10 [MillMax 0965-0-15-20-80-14-11-0](#) spring pins and a laser-etched plastic cover mounted on top of the spring pins and PCB to provide a seamless cover over the interface block cutouts on the module base.

The prototype interface block supports a single LVDS data lane with bi-directional data and clock differential pairs. Figure 3.9 and Table 3.13 details the pinouts of the prototype interface block.

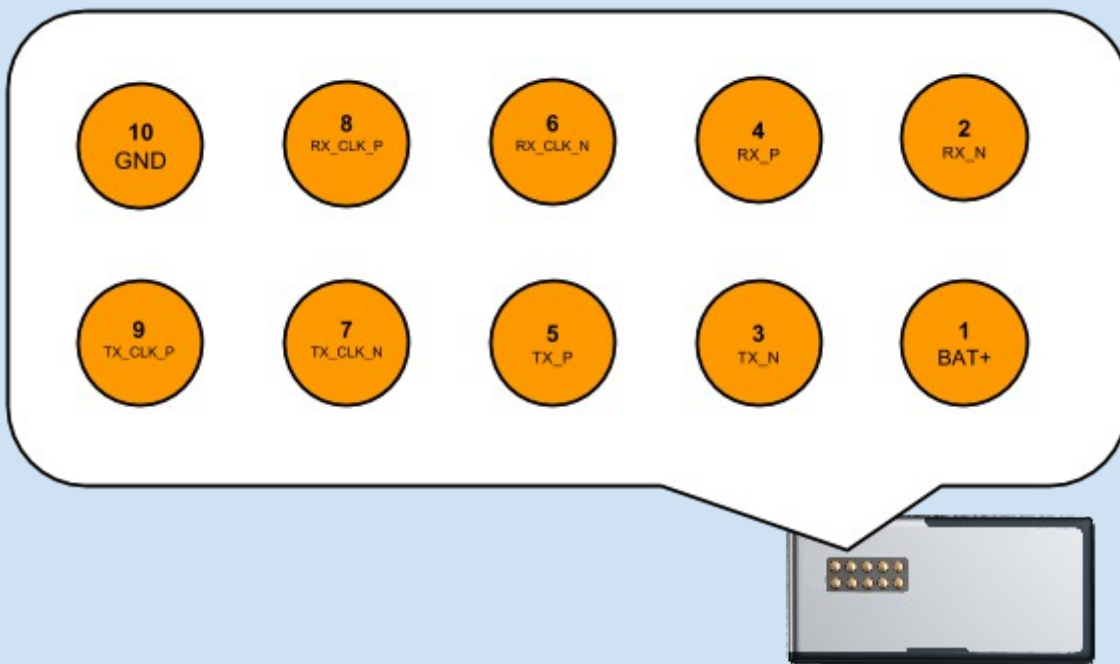


Figure 3.9 - Prototype Interface Block Pinout

Table 3.13 - Prototype Interface Block Pinout Descriptions

#	Pin/Pad Label	Description
1	+BATT	Power
2	RX_N	Data Receive Negative
3	TX_N	Data Transmit Negative
4	RX_P	Data Receive Positive
5	TX_P	Data Transmit Positive
6	RX_CLK_N	Receive Clock Negative
7	TX_CLK_N	Clock Transmit Negative
8	RX_CLK_P	Receive Clock Positive
9	TX_CLK_P	Transmit Clock Positive
10	GND	Ground

3.2.2. Electro-permanent magnets (EPM)

The EPMs provide a low-power and user-controllable method to securely attach modules to slots in the endoskeleton. The EPM has two selectable states: the attach state and release state, corresponding to high and low levels of magnetic force. Electrical power is needed to switch between the two states only; the EPMs require no sustained electrical power to maintain either state.

EPMs in the attach state provide sufficient magnetic force to secure modules into their slots on the endo throughout all nominal usage scenarios. EPMs in the release state provide a residual magnetic force to prevent modules from falling out unless the user deliberately removes the module from the endo. Users should be able to remove modules with minimal effort when the EPM is in the release state.

3.2.2.1. Rear Module EPM - Specification

The 1x1 and 2x2 modules each use a single EPM. 1x2 modules use two EPMs, one for each valid module insertion direction. Each EPM must provide a minimum holding force of 30 N in the attach state, and 3 N in the release state. This force must be applied in the insertion direction of the module, i.e., in the X direction. The EPM attachment surfaces on the endo are made from Hiperc-50 alloy to provide enhanced magnetic holding force. EPMs control functions are defined in the System-Level Functions section.

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

The EPMs must be shaped to conform to the pebble-like form expressed by the Ara module design language. They must install flush with the module base to form a smooth outer surface.

3.2.2.2. Rear Module EPM - Prototype Reference Implementation

Figure 3.10 shows an EPM used in the prototype. The front curved face mates with the Hiperc-50 attachment surface on the endo spine. The EPM is made from three parallel sections. The N42SH magnets at the front are magnetized parallel to the long axis of the device, alternating magnetization direction from section to section. The Alnico magnets in the back can have their magnetization direction reversed by passing a current through the coils. When the Alnico magnets are magnetized opposite to the N42SH magnets, the holding force is low; this is the release state. When the Alnico and N42SH magnets are magnetized together, the holding force is larger; this is the attach state.

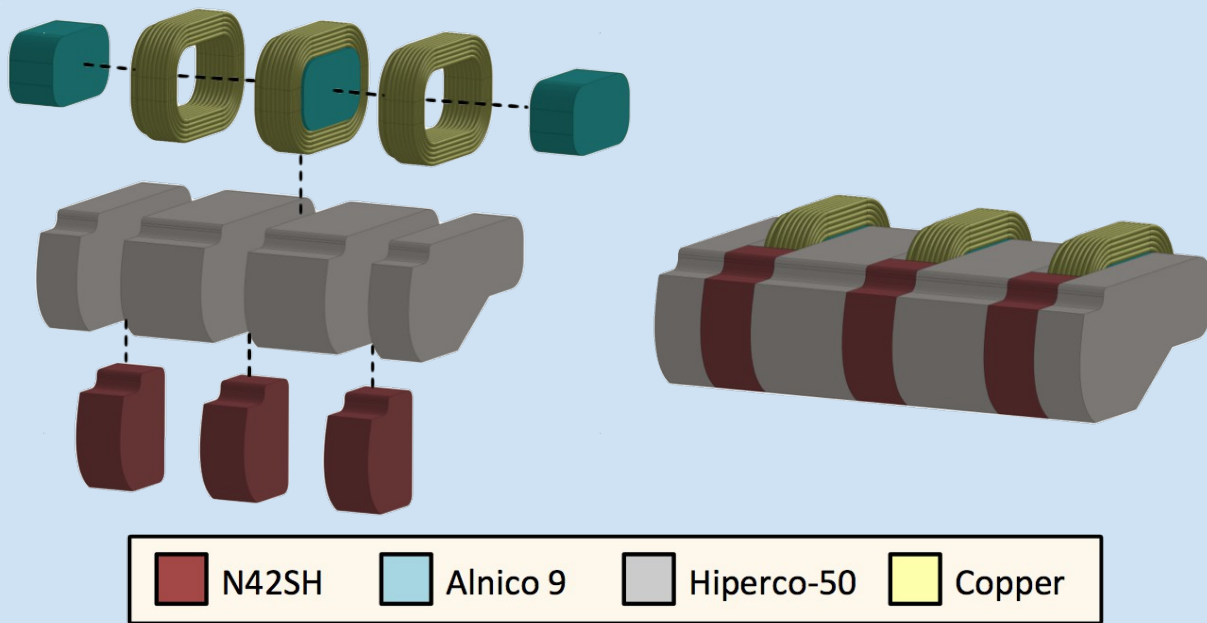


Figure 3.10 - Rear Module EPM Exploded View

To generate a switching pulse, the EPM driver circuit discharges a 10 μ F capacitor charged to 28 V across the coil for 22 μ s, resulting in a very short but high-current (10 A) pulse. More than one pulse across each coil is needed to reach full strength. Figure 3.11 below shows the pulse sequence to switch from the release state to the attach state. In the prototype reference implementation, $L = 22 \mu$ s, $S = 10$ ms, and $N = 4$. A sequence with reversed polarity switches from attach to release states.

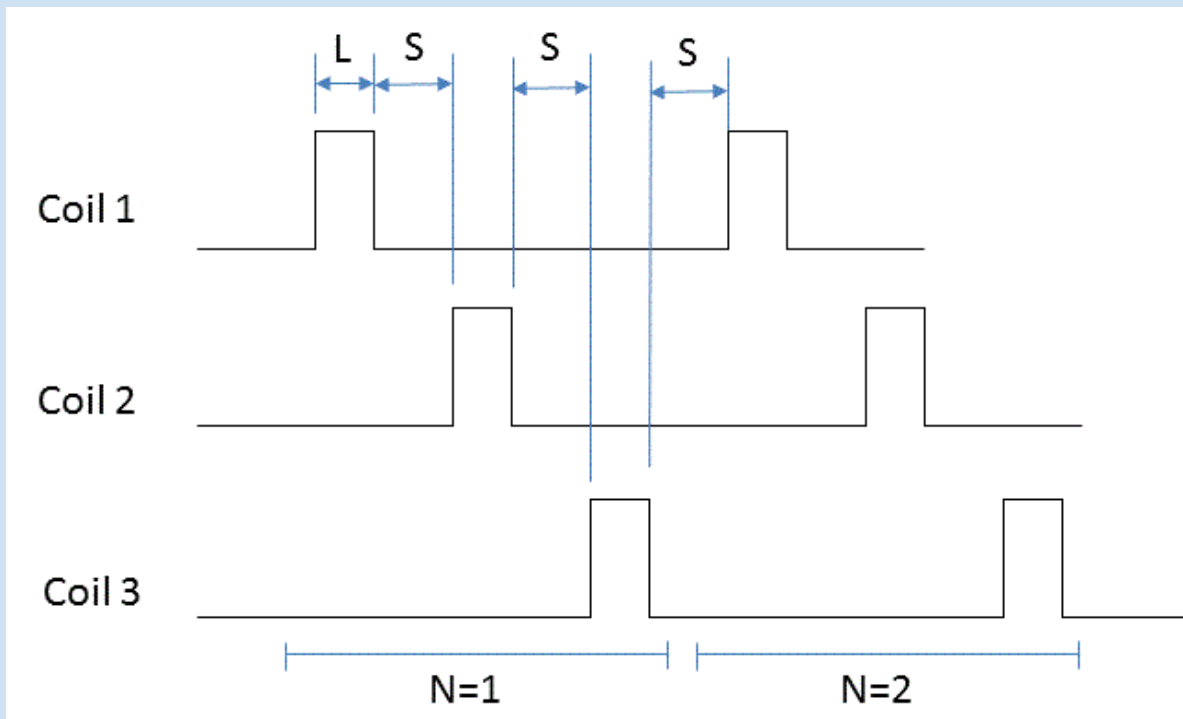


Figure 3.11 - Prototype EPM Drive Timing

Table 3.14 provides relative paths to design artifacts to create prototype rear module EPMs for 1x2 and 2x2 modules.

Table 3.14 - Rear Module EPM Prototype Reference Implementation

Component	Design Artifacts
1x2 Rear Module EPM	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\1x2\Prototype1x2EPM\CADFiles EDA Files for Driver Circuits: ReferenceMaterials\1x2\Prototype1x2ModuleTemplate\EDAFiles
2x2 Rear Module EPM	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\2x2\Prototype2x2EPM\CADFiles EDA Files for Driver Circuits: ReferenceMaterials\2x2\Prototype2x2ModuleTemplate\EDAFiles

There are no known commercial sources for miniature EPMs, such as those needed for Ara modules. Developers are encouraged to fabricate their own EPMs in accordance with the specification above. The Project Ara team is also working to establish a supply base for EPMs. In the near term, limited quantities will be available from AQS and NK Labs. Contact

information for both companies can be found in the Support section at the beginning of the MDK.

3.2.2.3. Front Module Attachment - Specification

Front modules attach to the endo with an EPM and ball-spring plunger assembly. The ball-spring plunger aligns front modules to their slots in the X direction (side-to-side) and prevents modules from sliding out. Once in place, an EPM secures the module in its place until deactivated by the user.

3.2.2.4. Front Module Attachment - Prototype Reference Implementation

Front modules on the prototype use a single ball-spring plunger assembly without an associated EPM. Figure 3.12 shows a model of the ball-spring plunger in relation to the Media Bar Module. The assembly uses a 2 mm long compression spring with a 1.2 mm outer diameter.

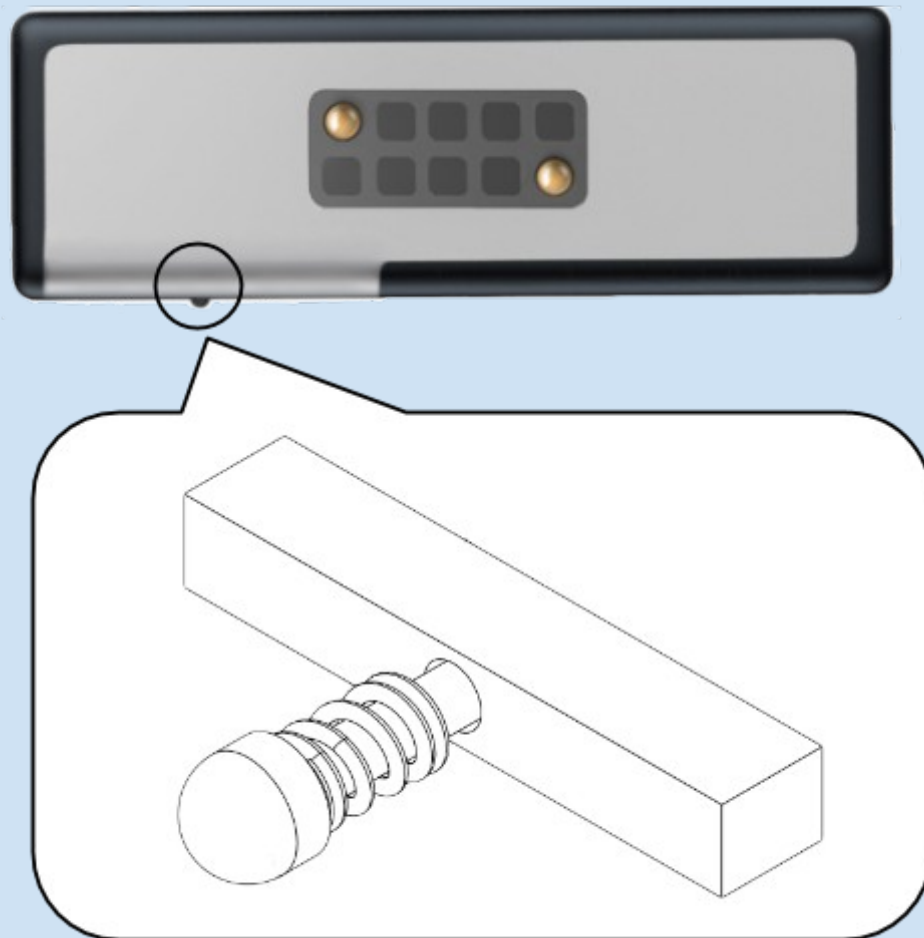


Figure 3.12 - Prototype Ball-Spring Plunger Assembly

Table 3.15 provides relative paths to design artifacts to create a prototype ball-spring plunger assembly.

Table 3.15 - Prototype Ball-Spring Plunger Assembly Reference Implementation	
Component	Design Artifacts
Ball-Spring Plunger Assembly	<ul style="list-style-type: none"> 3D Models and Drawings: ReferenceMaterials\Class(Medium)\PrototypeBall-SpringPlungerAssembly\CADFiles\

4. Power

This section defines power specifications and reference implementations for Ara modules. The Ara power architecture is designed to accommodate the unique and flexible nature of the platform. Users of an Ara phone will be able to power their device with one or multiple batteries; they will be able to swap a depleted battery with a fresh one, without powering off their phone; they will be able to charge one or more batteries in their phone from one or multiple charging devices. The design of the power architecture enables all of these use cases and provides a flexible power platform for new applications.

4.1. Power - Specification

Figure 4.1 shows a high-level view of the Ara power architecture. A common power bus is central to the architecture. The power bus is held between 3.0 and 5.5 V. The power bus is distributed through the endo frame to each interface block. A power manager in the endo optimizes power consumption, reduces current leakages, and provides protection against overcurrents. The endo also has a small battery or capacitor bank, and associated charge controller. The power stored in the endo provides power to the phone during brief periods when the battery and/or charger modules are being hot-swapped or reconfigured and no other power source is available to keep the device on.

Ara modules fall into three categories in relation to the power architecture: power consumers, charging modules, and power storage modules (batteries, fuel cells, etc.). Modules can change categories throughout their usage life. For instance, a battery module may have a built-in charging port. Such a module would serve as a power consumer when being charged from the bus (e.g., if another module's charging port is active), a charging module when powered through its charging port, and a power storage device when functioning as a regular battery. Figure 4.1 also illustrates a simplified view of these three module types. The following subsections describe each of the types.

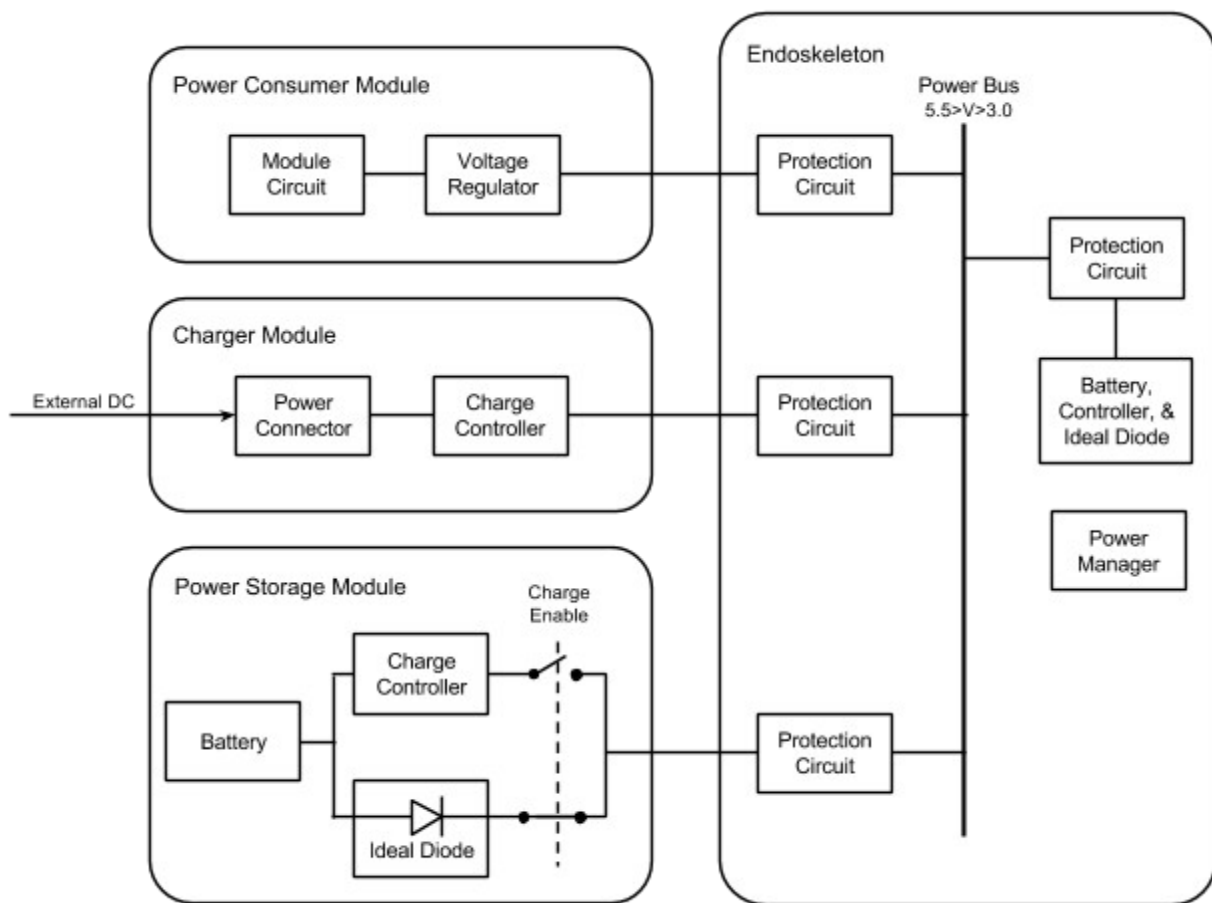


Figure 4.1 - Power Architecture

4.1.1. Power Consumer Module - Specification

At the top of Figure 4.1 is a power consumer module, which is the simplest in its operation. The power consumer module draws power directly from the power bus, or more commonly, voltage regulator(s) steps down and regulates the voltage from the bus for various circuits within the module.

4.1.2. Charger Module - Specification

Charging modules can supply externally derived power to the power bus. Charging modules may draw a small amount of current from the power bus during EPM activation and the module enumeration process, but current flows nominally in one direction, from the module to the bus. A traditional DC charger module is shown in Figure 4.1.

4.1.3. Power Storage Module - Specification

Power storage modules primarily include battery modules, but other power storage devices are envisioned. Power storage modules can supply power to the rest of the system or draw power from the bus for recharging.

Battery modules supply power to the power bus by default. Ideal diode circuits prevent current flow into a battery, and, therefore the power bus voltage is equal to the highest voltage presented by any of the batteries connected to the power bus. When a charger module is connected to the system and ready to deliver charging current to the batteries, it signals the supervisory controller in the endo (central power management system) via UniPro messaging. The supervisory controller then signals battery modules to cut off power delivery to the power bus and activate their charging circuits to draw power from the bus. The supervisory controller periodically polls battery modules to report self diagnostics such as state of charge and uses this information in its power management routines.

4.2. Power - Reference Implementation

Reference implementations in the subsequent sections demonstrate compliant designs for power consumer, charger, and power storage modules.

4.2.1. LED Array Module - Prototype Reference Implementation

Table 4.1 provides relative paths to design artifacts to create an example of a power consumer module in a 1x2 module template. The LED Array module is a simple power consumer module that uses voltage regulators to modulate the color of the LEDs.

Table 4.1 - 1x2 Prototype LED Array Module Reference Implementation

Component	Design Artifacts
LED Array Module	<ul style="list-style-type: none">3D Models and Drawings: ReferenceMaterials\1x2\PrototypeLEDArrayModule\CADFilesEDA File and Outputs: ReferenceMaterials\1x2\PrototypeLEDArrayModule\EDAFiles

4.2.2. USB Charger Module - Prototype Reference Implementation

Table 4.2 provides relative paths to design artifacts to create a USB Charger Module in a 1x2 module template. The USB Charger Module uses a standard USB Micro-B socket. An external power supplies USB compliant DC power.

Table 4.2 - 1x2 Prototype USB Charger Module Reference Implementation

Component	Design Artifacts
USB Charger Module	<ul style="list-style-type: none">3D Models and Drawings: ReferenceMaterials\1x2\PrototypeUSBChargerModule\CADFilesEDA File and Outputs: ReferenceMaterials\1x2\PrototypeUSBChargerModule\EDAFiles

4.2.3. Battery Module - Prototype Reference Implementation

Table 4.3 provides relative paths to design artifacts to create a Battery Module in a 2x2 module template. The Battery Module uses a lithium-ion battery pack for power storage.

Table 4.3 - 2x2 Prototype Battery Module Reference Implementation

Component	Design Artifacts
Battery Module	<ul style="list-style-type: none">3D Models and Drawings: ReferenceMaterials\2x2\PrototypeBatteryModule\CADFilesEDA File and Outputs: ReferenceMaterials\2x2\PrototypeBatteryModule\EDAFiles

5. Network Stack

This section defines the Ara network protocol stack.

5.1. Module Communication Physical View

Figure 5.1 illustrates the interface that enable module to module communication on an Ara phone. The network architecture centers around the MIPI M-PHY and UniPro protocols. M-PHY and UniPro are specifically designed for mobile devices with advanced features to provide high throughput with low power consumption. These protocols create a packet switched network on an Ara phone and enable any module to communicate with any other module through a switch in the endo.

Figure 5.1 shows 3 types of modules: An AP module with native M-PHY and UniPro support, a Display Module with a DSI-2 interface, which is a UniPro-compatible MIPI standard (other UniPro-compatible MIPI standards include CSI-3 for cameras and UFS for mass storage devices), and generic peripheral modules (a Wi-Fi Module is shown in Figure 5.1). Peripheral modules without native M-PHY and UniPro support may utilize a General Purpose Bridge ASIC that converts between UniPro and several standard chip-to-chip protocols including SDIO, USB HSIC, UART, I2C, I2S, and GPIO (see M-PHY reference implementation section for details on the General Purpose Bridge ASIC). Up to 4 M-PHY lanes per module are available depending on the number of interface blocks (2 M-PHY lanes per interface block).

The modules in Figure 5.1 connects to a 14-port UniPro switch in the endo over a contactless capacitive interface to be described in the subsequent section. A supervisory controller in the endo also connects to the UniPro switch on UniPro CPort 0.

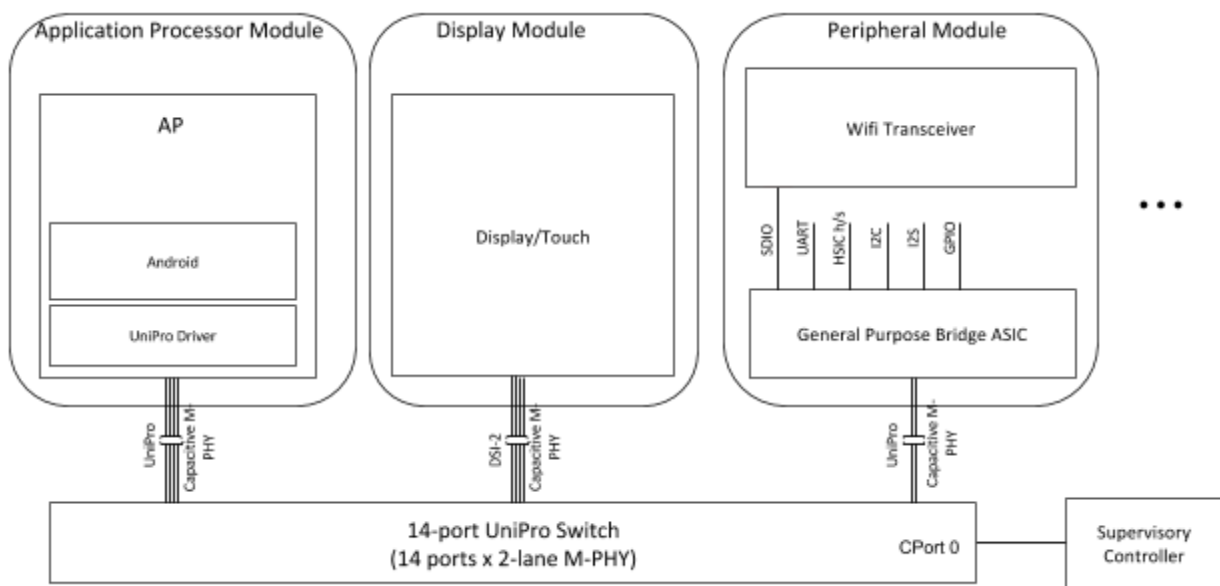


Figure 5.1 - Module to Module Communication (with Native UniPro Support and Bridge ASICs)

Figure 5.2 describes an intermediate solution between the current prototype system and the objective vision. This architecture adds a UniPro AP/Display/Camera Bridge ASIC (AP Bridge for short) to convert between high-speed and high-bandwidth device interfaces in AP, display, and camera devices to UniPro, enabling a less efficient but more near-term solution. The M-PHY reference implementation section provides more details on the AP Bridge ASIC.

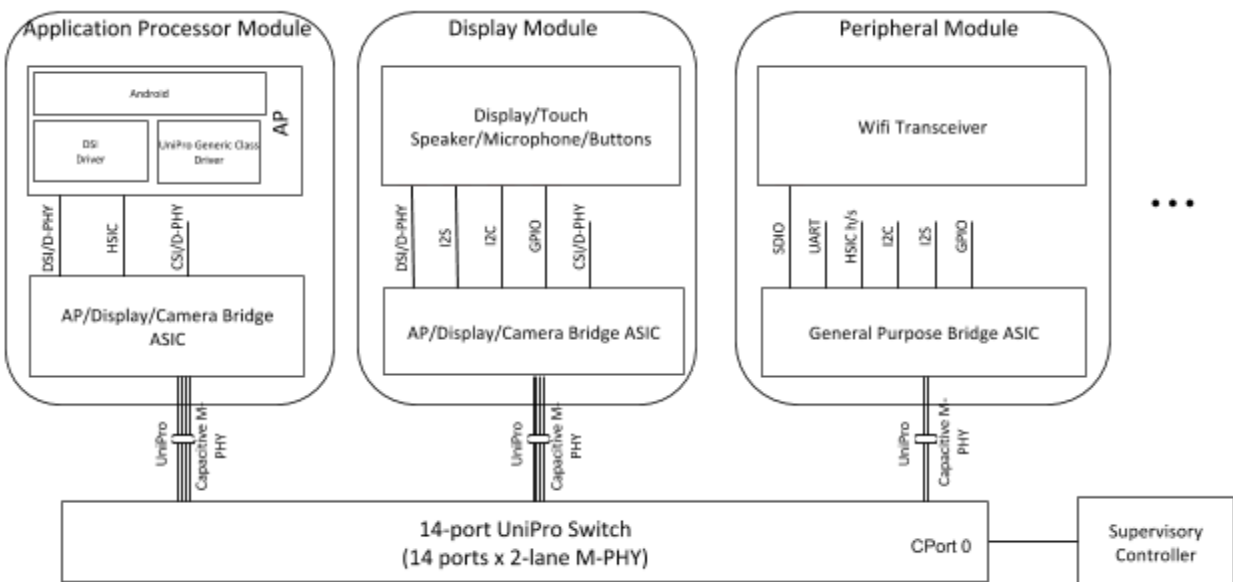


Figure 5.2 - Module to Module Communication (with UniPro Bridge ASICs)

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

The prototype differs from the objective system in two key aspects, as illustrated in Figure 5.3. First, the prototype uses LVDS as the physical layer and spring pins on the interface block instead of M-PHY and contactless capacitive data transfer. Second, in lieu of devices with native UniPro support, an FPGA converts several common chip-to-chip protocols (I2C, I2S, SDIO, GPIO, and DSI) to UniPro and performs the reverse operation on the other end, at the AP. The FPGAs allow components that use one of these tunneled protocols to communicate with an AP without the need to implement UniPro (or LVDS). UniPro is completely hidden from the perspective of these devices. This tunnelling scheme enables development of prototype Ara modules in parallel with development of ASICs with UniPro and M-PHY support.

The module templates in the Mechanical section of the MDK describe the FPGA and associated circuitry to instantiate a module that can communicate over one of the provided tunneled protocols.

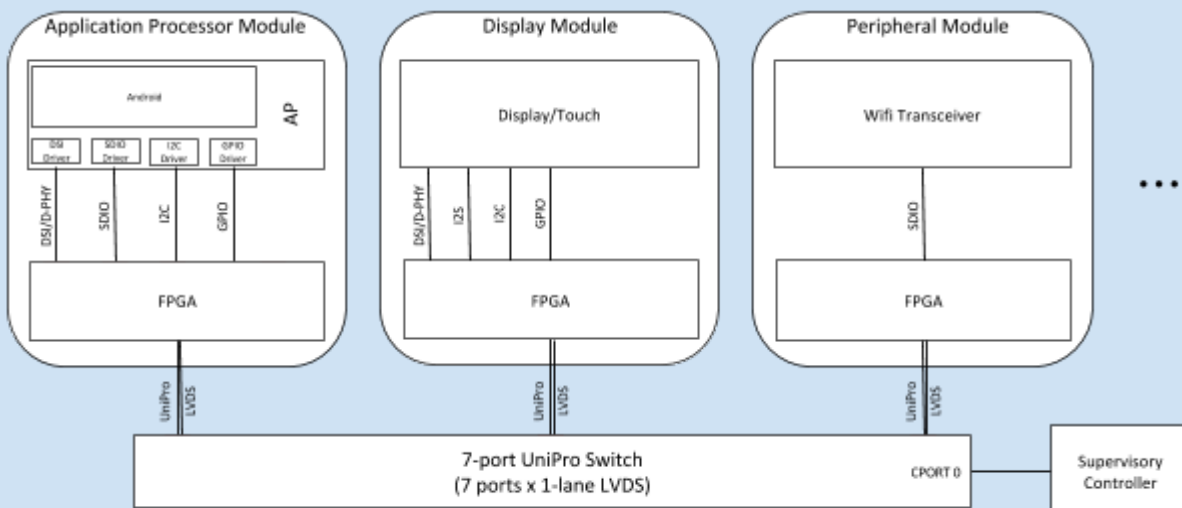


Figure 5.3 - Prototype Module to Module Communication

5.2. Network Stack

The Ara network protocol stack is summarized in Table 5.1. The Ara network is packet-switched and based on the MIPI M-PHY and UniPro standards. The network enables high data rate communication between modules with low pin count and low power consumption.

Table 5.1 - Ara Network Stack

OSI Layer	Ara Implementation
Layers 5-7	UniPro class driver or Bridge ASIC and custom userspace driver
Layers 2-4	MIPI UniPro v1.6
Layer 1	MIPI M-PHY v3.0 with capacitive media converter

The prototype network stack differs from the objective system in several aspects. Layer 1 is a standard LVDS implementation and in Layers 2-4, UniPro is implemented but is hidden from the module point of view. In place of native UniPro, the prototype tunnels several common chip-to-chip interfaces including DSI, I2C, I2S, GPIO, and SDIO through a custom FPGA provided in the prototype module template. The Software section describes Android features that expose these interfaces to higher layers of the stack.

Table 5.2 - Prototype Ara Network Stack

OSI Layer	Prototype Ara Implementation
Layers 5-7	Android platform with support for prototype FPGA's interfaces
Layers 2-4	DSI, I2C, I2S, SDIO, DSI, and GPIO tunneled over UniPro
Layer 1	LVDS

5.2.1. Layer 1

Layer 1 includes the capacitive media converter and M-PHY.

5.2.1.1. Layer 1 Capacitive Media Converter - Specification

The capacitive media converter allows high-speed and contactless transmission of data between modules and the endoskeleton. The capacitive media converter (which will be integrated into the bridge IC) removes the need for a data connector, saving space and cost, and increasing durability.

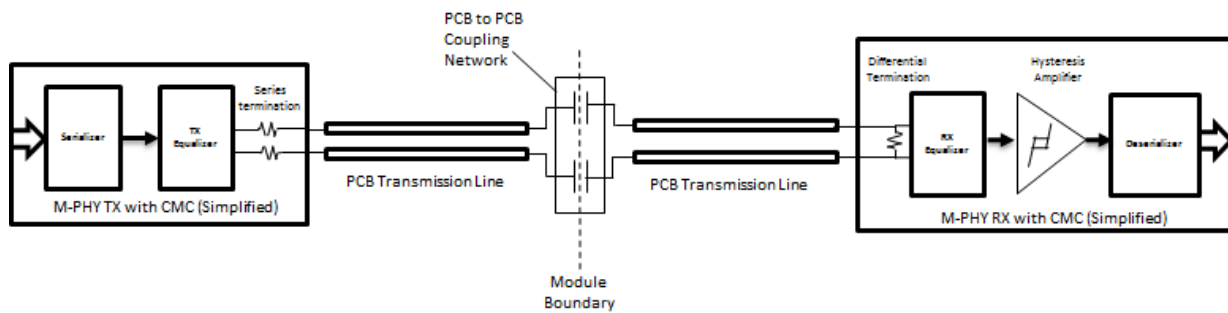


Figure 5.4 - Capacitive Media Converter

Figure 5.4 presents a high-level view of the capacitive media converter. A microstrip structure on the bottom layers of the module PCB couples the data signal to a corresponding structure in the endoskeleton. Coupling occurs both by capacitance, at high frequencies, and by mutual inductance, at lower frequencies. The channel is high-pass. To allow low-speed data transmission, the DC level of the signal is restored in the receiver by a hysteresis amplifier.

The capacitive media converter will operate over the 0.1 mm (+/- 0.05 mm) air-gap between the module and endo. It will support all M-PHY transmission rates and modes implemented by the Ara network (see M-PHY specifications section). A detailed specification containing electrical requirements for the transmitter and receiver, and the geometric tolerances for the microstrip coupling network, will be released in a future version of the MDK.

5.2.1.2. Layer 1 Capacitive Media Converter - Reference Implementation

A reference implementation for the capacitive media converter is under development.

5.2.1.3. Layer 1 MIPI M-PHY - Specification

The Ara network stack uses [MIPI M-PHY](#) (V3.0 26-July-2013) at Layer 1. M-PHY is a serial interface that supports multiple transmission rates for high and low bandwidth applications. M-PHY uses 8b/10b encoding and an embedded clock signal. M-PHY also has multiple power saving modes to support low power embedded devices.

Programmable attributes of the M-PHY standard have not been fully specified for the Ara platform. The current plan is to support all three High Speed GEARs up to HS-GEAR 3 (at 3A/3B Bandwidth and 4.99/5.82 Gbps per lane). Pulse Width Modulation (PWM) will be supported only up to PWM-GEAR4 (at 1/2/3/4 Bandwidth and 9/18/36/72 Mbps per lane).

5.2.1.4. Layer 1 MIPI M-PHY - Reference Implementation

A reference implementation for M-PHY/UniPro network bridge(s) is under development. See the Network Stack Hardware section below for additional details. M-PHY IP blocks are also available from multiple sources.

5.2.1.5. Layer 1 LVDS - Prototype Specification

The prototype platform uses [LVDS](#) with AC coupled at 400 mV. The encoding scheme is double data-rate (DDR) supporting up to 800 Mbps. High impedance signalling is used for wake and detect signalling at 0 V and 3.3 V DC offsets. Termination resistors must be spaced at 100 ohm for line impedance matching.

5.2.1.6. Layer 1 LVDS - Prototype Reference Implementation

The module templates include EDA files and schematics that detail the pinouts of the prototype FPGA and interface block with respect to the LVDS signals.

5.2.2. Layers 2, 3, 4

5.2.2.1. Layers 2, 3, 4 MIPI UniPro - Specification

The Ara network stack uses [MIPI UniPro](#) (V1.6 6-August-2013) at the data link, transport, and network layers. UniPro is designed specifically for mobile applications and enables a lightweight, low-latency packet-switched network between modules on an Ara phone. A UniPro switch in the endo routes packets between modules and provides quality of service guarantees to system-level functions and modules.

The UniPro data link layer (Layer 2) defines 2 traffic classes: TC0 and TC1. TC1 data frames are sent before TC0. Data packets from all modules use TC0 while system-level interrupts use TC1. A credit based flow control mechanism enables receivers to tell the transmitter how much buffer space is available and pause the transmission to avoid a receive buffer overflow. An incorrect frame checksum at Layer 2 triggers an automatic retransmission.

The UniPro network layer (Layer 3) addressing scheme is tied to the [interface blocks](#) on the endo. A unique identification number is assigned to each interface block on the endo frame. The interface block ID is assigned by ISO convention, starting from the rear of the endo, from the left and top, moving in the counterclockwise direction, and ending at the front of the endo. Since the interface block locations are the same regardless of rib placement, they can be statically assigned. The result is shown in Figure 5.5.

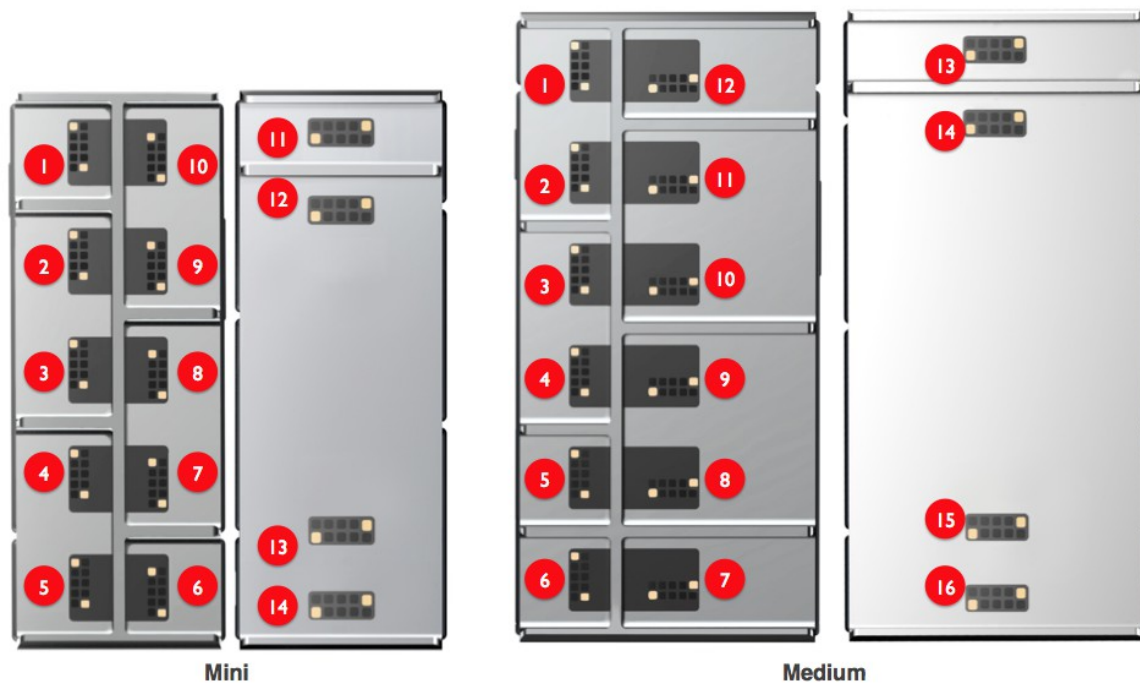


Figure 5.5 - Interface Block Unique IDs

The UniPro transport layer (Layer 4) handles connections between modules using logical data streams. UniPro uses CPort identifiers, which are equivalent to port numbers used in TCP or UDP. A pair of CPorts, one on each of two modules, can form a bi-directional connection. Each module may have multiple CPorts, which enables a module to simultaneously connect and communicate with multiple modules on an Ara device.

5.2.2.2. Layers 2, 3, 4 UniPro - Reference Implementation

A reference implementation for M-PHY/UniPro network bridge(s) is under development. See the Network Stack Hardware section below for additional details. UniPro controller IP blocks are also available from multiple sources.

5.2.2.3. Layers 2, 3, 4 UniPro Tunnel - Prototype Specification

The prototype system tunnels several common chip-to-chip interfaces through UniPro. The tunneled protocols include DSI, I2C, I2S, GPIO, and SDIO. These protocols are selected to facilitate development of various prototype modules in parallel with the UniPro ASIC development.

5.2.2.4. Layers 2, 3, 4 - Prototype Reference Implementation

The prototype FPGA implements the following interface tunnels. Pinouts for each tunneled interface from the FPGA are detailed in the EDA files and schematics in the prototype module template section. The sections below describe any deviations or specific parameters implemented for the standards.

5.2.2.4.1. I2C Tunnel - Prototype Reference Implementation

The touchscreen controller in the prototype Display Module, the audio CODEC in the Display and Media Bar Modules, the Thermal Imager Module, and the Pulse Oximeter Module all use the I2C tunnel. The I2C tunnel supports clock stretching and repeated starts.

5.2.2.4.2. DSI Tunnel - Prototype Reference Implementation

The prototype Display Module uses the DSI tunnel. The DSI tunnel supports four lanes plus a clock. It is HS mode only, with no support for LP mode or bus turnarounds. The HS clock runs at 108 MHz. The display is a 1280x720 display, 24 bit color, command mode, and is updated by the AP at an average rate of 15 Hz.

5.2.2.4.3. GPIO Tunnel - Prototype Reference Implementation

The LED Array Module and the Display Module use the GPIO tunnel. A bank of GPIO lines is provided in each direction, with 100 microsecond latency.

5.2.2.4.4. I2S Tunnel - Prototype Reference Implementation

The audio CODEC in the Display and Media Bar Modules use the I2S tunnel. The AP serves as the I2S master and each audio CODEC is a slave. The audio is in DSP format, with a 48 kHz sampling clock and 64 bits per frame. Data is valid on the rising edge of the clock. The frame sync line goes high for one clock cycle per frame to denote the start of the frame. The frame sync line goes high for one rising edge of the clock. The data for the left stereo channel starts (MSB first, 16-bit) on the next clock edge, the right stereo channel data follows, and ends in 32 bits of zeros. The I2S tunnel is bidirectional, and supports simultaneous audio transfer in each direction.

5.2.2.4.5. SDIO Tunnel - Prototype Reference Implementation

The prototype Wi-Fi Module use the SDIO tunnel to connect their respective Wi-Fi chipsets to the AP.

5.2.3. Layers 5+

5.2.3.1. Layers 5+ - Specification

Figure 5.6 shows the protocols at Layer 5 and above for two classes of devices. The first class of devices has native UniPro support. These devices will use class drivers built on the UniPro

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

stack. Examples devices include cameras that utilize MIPI CSI-3 for video streaming or mass storage devices that use UFS. New device class drivers built on UniPro for audio streaming, wireless communications and others are also under development.

Devices without native UniPro support will utilize a General Purpose (GP) Bridge ASIC to communicate over the UniPro network. A remote host controller (RHC) in the GP Bridge ASIC will drive communications with devices over their native interface such as I2C or HSIC. Layer 5 commands and arguments will be sent over the network as UniPro packets encapsulating remote procedure calls. The Adroid services and Hardware Abstraction Libraries (HALs) communicate with userspace drivers with the protocols' native commands such as i2c_rdwr for I2C and USB Request Blocks (URB) for HSIC. On the AP side, these commands and data will be encapsulated as UniPro packets in the AP Bridge ASIC and transferred over the UniPro network to the endpoint. In the GP Bridge ASIC, these UniPro packets are unpacked and delivered as remote procedure calls to the remote host controller, which drives the native interface.

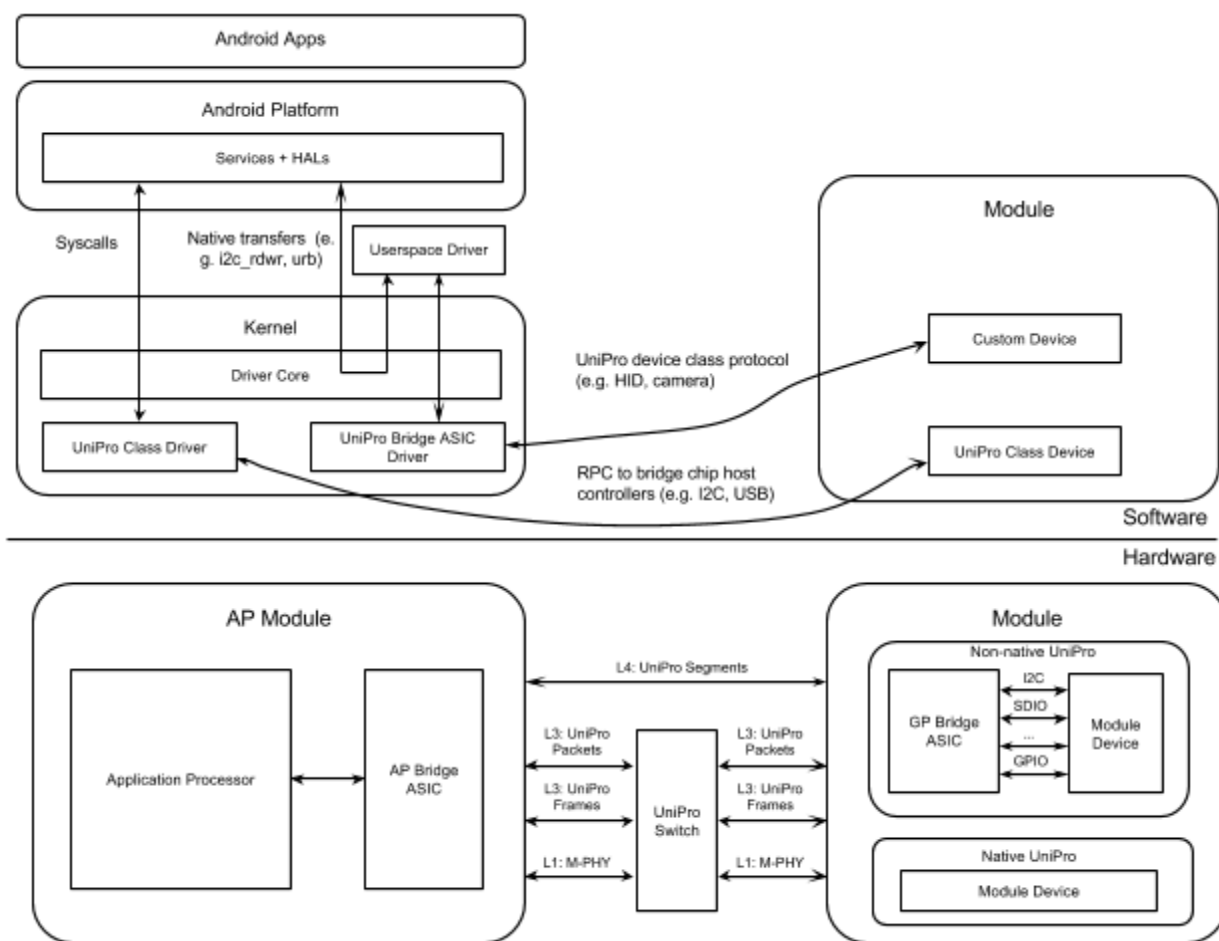


Figure 5.6 - Network Stack over Software and Hardware

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

5.2.3.2. Layers 5+ - Prototype Reference Implementation

The prototype development platform is designed to allow module developers to reuse existing standards and software for session, presentation, and application layer management. The software platform is a modified Android source release with support for the prototype FPGA that tunnels legacy protocols such as I2C, SDIO, GPIO, DSI, and I2S over UniPro.

5.2.4. Network Stack Hardware - Reference Implementation

Peripherals can interface with the Ara network either through a native UniPro/M-PHY protocol implementation or by utilizing a bridge chip. Two such UniPro bridge chips are under development by Toshiba Semiconductor and Storage Products Company: a UniPro AP/Camera/Display Bridge and a UniPro General Purpose (GP) Bridge. Table 5.3 provides some preliminary specifications for the two bridge ASICs.

Table 5.3 - Toshiba UniPro Bridge ASIC Specifications

ASIC	UniPro Interfaces	Peripheral Interfaces	Chip Information
AP Bridge	2-bi-directional MIPI M-PHY v3.0 Type-1 M-MPORTs with maximum speed at HS-G3b mode (5.8 Gbps/lane, 11.6 Gbps/M-PORT) and capacitive media converter	2 MIPI CSI-2 links (4 D-PHY lanes @ 1Gbps/lane) 2 MIPI DSI links (4 D-PHY @ 1Gbps/lane) USB HSIC v1.0 host I2C Master with normal, fast, and fast mode plus I2S audio with max 48 kHz stereo 10+ GPIO lines EPM control lines	Power: TBD Size: TBD Pin-count: TBD
GP Bridge	2-bi-directional MIPI M-PHY v3.0 Type-1 M-MPORTs with maximum speed at HS-G3b mode (5.8 Gbps/lane, 11.6 Gbps/M-PORT) and capacitive media converter	USB HSIC v1.0 host SDIO v2 host controller SPI Master I2C Master with normal, fast, and fast mode plus I2S audio with max 48 kHz stereo 2 UARTs 10+ GPIO lines EPM Control lines	Power: TBD Size: TBD Pin-count: TBD

5.2.5. Network Stack Hardware - Prototype Reference Implementation

The prototype system uses a Lattice Semiconductors FPGA with custom bitstreams to implement the network stack. In the near term, limited quantities will be available from AQS. Contact information can be found in the Support section at the beginning of the MDK.

6. Software Stack

Figure 6.1 shows an abstracted view of the Ara software stack. The Ara software stack is predicated on the attachment of a single Application Processor (AP) Module running Android. (Support for multiple AP modules may be included in a future release of the MDK.) From the perspective of an Ara phone's Linux kernel, there are two types of modules: those which conform to a particular device class (device classes such as cameras, displays, human interface devices, etc., which will be defined in a future release of this document), and novel, unique, or otherwise special-purpose modules, which do not belong to any device class.

A future release of the MDK will be accompanied by a set of device class definitions, UniPro-based communication protocols, and associated kernel drivers. Devices without in-kernel device class drivers are unlikely to have native UniPro interfaces, and will require UniPro bridge chips as shown in Figure 5.2 in the Network section to convert between their native interfaces and UniPro. These devices will communicate with Android through a UniPro Bridge chip driver in the kernel and a developer-supplied userspace driver. Figure 6.1 shows the software interfaces and interactions for modules with and without existing in-kernel device class drivers.

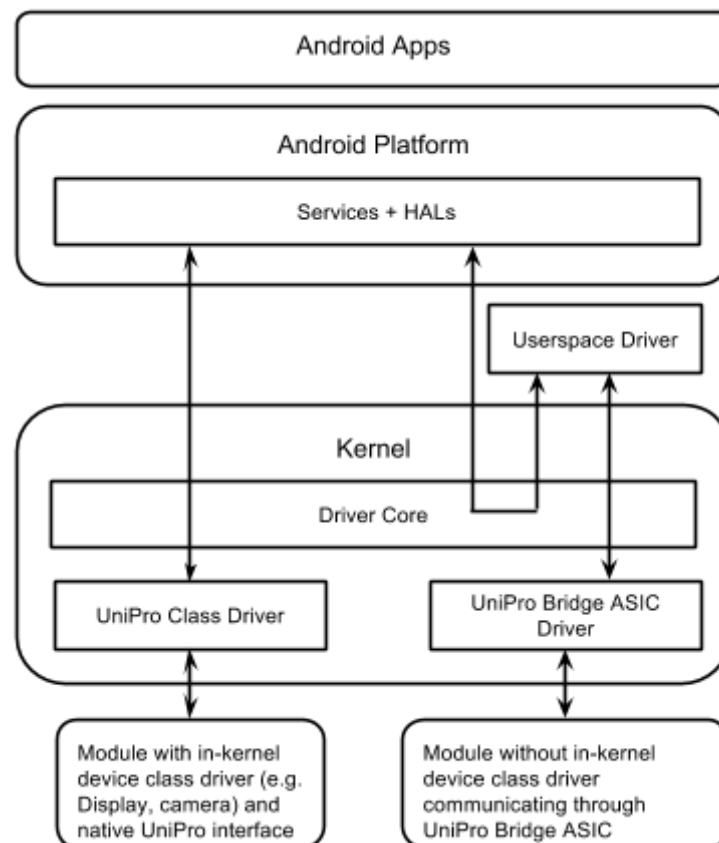


Figure 6.1 - Software Stack

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

The present prototype platform software is a modified version of the Linaro 13.09 Android release, targeting a TI OMAP 4460 AP. It lacks any provisions for UniPro support, as envisioned for the objective platform, and instead relies on a static set of custom device drivers for all prototype modules. Although this approach is not scalable to support a commercial release of the Ara platform, it is a stop-gap approach that can support module hardware development until UniPro class driver support can be developed and mainlined in Android.

6.1. Kernel Drivers

Linux device drivers provide low-level hardware support on Android devices. On a standard smartphone, the available hardware is fixed, allowing device manufacturers to pre-select a set of device drivers. On an Ara phone, however, almost all of the hardware which is normally an immutable part of a smartphone's design is designed to be user replaceable and hot-pluggable.

6.1.1. Kernel Drivers - Specification

To enable hot-plug support and provide a framework that enables arbitrary future extensions, two driver classes will be developed, as previously shown in Figure 6.1: native UniPro-based class drivers and UniPro Bridge ASIC-based user mode drivers. UniPro class drivers will support a range of module devices with in-kernel device drivers that will be part of the base Ara software release.

The UniPro Bridge ASIC drivers will allow a module device without native UniPro interface or in-kernel device driver to communicate with Android through the Linux driver core and a developer-supplied userspace driver. The design of this interface is influenced by the [Filesystem in Userspace \(FUSE\) project](#).

6.1.2. Kernel Drivers - Prototype Reference Implementation

Since UniPro software support is not yet mature, for prototype module development, module developers must create custom kernel images with driver support for their modules built-in. Communication between these drivers and the modules is tunneled over the protocols defined in the Network Stack section. The prototype Android release includes a kernel tree in `kernel/linaro/pandaboard` relative to the Android source checkout root directory.

6.2. Android HALs

Hardware Abstraction Libraries (HAL) are implemented as shared libraries; their code runs within Android's system services. Unlike device drivers, HAL APIs are specified by Google and are updated with each release. Device manufacturers provide library binaries which implement

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

these interfaces in their devices' system images. These libraries are then subsequently loaded by the relevant system services as part of their initialization. Among other things, HALs provide an abstraction barrier between Android system services and the various device driver interfaces exposed by the kernel for different hardware peripherals those services must control.

6.2.1. Android HALs - Specification

Current HAL APIs assume a fixed hardware configuration and will require modifications to support hot-plugging features inherent to the Ara architecture. A future release will address these modifications.

6.2.2. HALs - Prototype Reference Implementation

Sample HAL sources are available online which provide example HAL implementations.

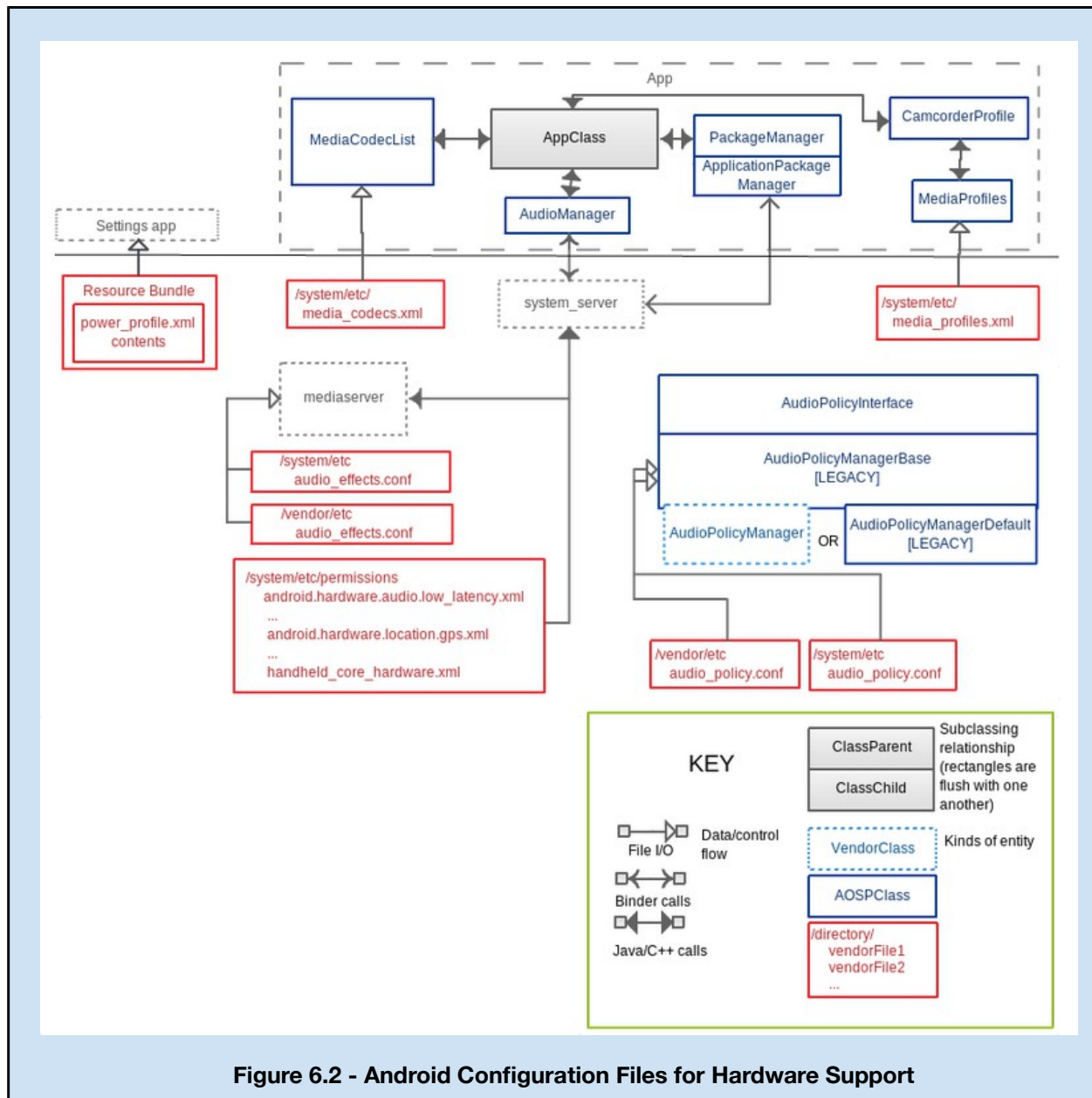
- Audio: [Wolfson Micro's `tinyHAL`](#)
- Sensors: [hardware/invensense/libensors](#)

6.3. Config Files - Prototype Reference Implementation

The Android platform requires various configuration files from device manufacturers. The [Android developers portal](#) describes these files for various types of hardware types. Figure 6.2 on the following page shows various system configuration files used in the prototype highlighted in red, along with various components of the Android platform which depend on them.

Example configuration files are available for various existing devices supported in publicly available Android Open Source Project releases.

- **Audio:** [device/samsung/tuna/audio_effects.conf](#),
[device/samsung/tuna/audio/audio_policy.conf](#)
- **System properties:** [device/samsung/toro/system.prop](#)
- **System initialization:** [device/samsung/tuna/init.tuna.rc](#)



6.4. Android Application

In general Android applications for the Ara platform are not in any significant respect different than traditional Android apps. Consequently, Ara devices are anticipated to be compatible with most standard apps.

6.4.1. Android Application - Specification

The [Android developers portal](http://developer.android.com/) describes Android application development. Additional guidance and reference materials are also readily available from many third party sources. App

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

developers are encouraged to pay attention to application stability and graceful behavior in light of dynamically varying availability of hardware resources (as with hot-plug, for instance).

6.4.2. Android Application - Reference Implementation

The Android source tree includes several pre-built applications available under `packages/apps/`, relative to the Android source checkout's root directory.

6.4.3. Android Application - Prototype Reference Implementation

The prototype includes a Thermal Imager Application and Pulse Oximeter Application located relative to the Android source checkout's root directory:

- Thermal Imager App: `packages/apps/arathermal`
- Pulse Oximeter App: `packages/apps/araplox`

These applications are also available as zipped and tarball files relative to MDK file at: `ReferenceMaterials\1x2\PrototypeThermalImagerModule\Software` and `ReferenceMaterials\1x2\PrototypePulseOximeterModule\Software`.

6.5. Prototype Development Use cases

This section describes several use cases module developers may encounter and includes references to existing support infrastructure provided by the prototype software platform.

6.5.1. I2C Devices

Some modules such as the prototype Thermal Imager and Pulse Oximeter Modules may require simple wrappers for chips communicating over the I2C protocol along with a special-purpose app that exposes the module's functionality. To support this use case, additional Android infrastructure is provided to interface with I2C devices from apps without any additional driver requirements.

The main Android Application Programming Interface (API) is the `android.hardware.I2cManager` class. For usage information, see the Javadoc comments in `frameworks/base/core/java/android/hardware/I2cManager.java` relative to the Android source checkout's root directory.

Modules communicating over I2C that require additional kernel drivers (e.g. touchscreens) are not covered by this case. For such cases, module developers should either use an existing driver in the kernel source tree or provide their own.

6.5.2. GPIO Devices

TBD. Like I2C devices, additional Android infrastructure will be provided to interface with GPIOs from apps without additional driver requirements.

6.5.3. I2S Devices

Modules using I2S components currently have no special-purpose support within the Ara software platform. Module developers should use an existing driver, or they may provide their own and integrate with the appropriate Android subsystems for their module's functionality as described in the [Android developers portal](#).

6.5.4. SDIO Devices

Modules using SDIO components currently have no special-purpose support within the Ara software platform. Module manufacturers should use an existing driver, or provide their own and integrate with the appropriate Android subsystems for their module's functionality as described in the [Android developers portal](#).

7. System-Level Functions

The following sections illustrate system-level functions performed by Ara modules. Modules will utilize the network and software stacks to exercise these functions. Modules interface with the supervisory controller in the endo to carry out these functions.

7.1. Module Detect

A feature of the Ara platform is that batteries and power buttons can be distributed among the modules. The platform uses out-of-band wake and detect signals to allow modules and the endo to power themselves off when appropriate.

The detect signals are used to communicate to a module and the endo when a module is inserted or removed. Power consuming modules naturally lose power when removed from the endo, so they automatically shut off. Modules with batteries must place themselves in a powered-down standby state when they lose the detect signal from the endo.

An Ara phone can power itself on and off by pressing buttons on a module. The wake signals are used to bring the phone out of a low-power standby state. The WAKE_TX signal is sent from a module to the supervisory controller when the power button is pressed. This brings the supervisory controller out of standby state, and the supervisory controller, in turn, sends WAKE_RX signals to each of the installed modules to turn them on.

The detect and wake signals are out-of-band signals superimposed onto the data lines of the UniPro network.

The prototype FPGA implements module detect and wake signals needed to communicate with the supervisory controller. These signals are transmitted as DC offsets onto the differential signals used for UniPro data communications, and are recovered with high input impedance buffers before the signals are AC coupled to their differential receivers.

The EDA files and schematics in the module templates details pinouts and signals from the FPGA for detect and wake signaling.

7.2. Enumeration

Enumeration allows the phone to identify and determine the capability of a module when it is plugged into the endo. For example, a Display Module will provide its vendor/manufacturer name, device class indicating the type of device driver it needs, screen resolution, and other performance identifier needed to operate the module. The enumeration process uses UniPro messaging.

When a module insertion event occurs, the UniPro switch on the endo signals an interrupt to the supervisory controller. The supervisory controller then establishes a low-speed UniPro link with the module and initializes the routing table with CPort logical addresses, device identification, M-PHY gearing, and bandwidth priority. The supervisory controller then sends a UniPro message to the other modules, including the Application Processor, through UniPro CPort 0 to announce the insertion event so appropriate drivers can be loaded.

Enumeration is not performed in the prototype platform.

7.3. EPM Control

EPM control addresses how EPM(s) in the module are switched between attach and release states. These actions are triggered by the user with an application interface. The application will graphically show the current state of modules and module slots in the phone including whether a module is present in the slot, the name identifier of a module, and the state of the module's EPM(s) (attach/detach). The user may then toggle the state of the EPM by touching on a specific module.

Once triggered by the user, the supervisory controller issues EPM control messages to the UniPro Bridge ASIC to activate the EPM driver circuits.

Users control the EPMs on the prototype with an EPM application located at `packages/apps/araepm` relative to the Android source checkout's root directory

The prototype FPGA includes an EPM core and driver circuits to generate microsecond-precise pulses to control the EPM(s).

7.4. Active Thermal Management

Active thermal management features protect the user and the phone from excessive heat buildup. Excessive heat generated from a module can cause injury to user and cause neighboring modules to fail. Temperature sensors in the endo and the modules provide measurements to the supervisory controller. If temperatures exceed predefined limits, modules may receive warnings, and in the worst case, power will be removed.

Active thermal management is not implemented in the prototype platform.

8. Environmental

This section provides environmental specifications for Ara modules. These specifications are designed to ensure modules can survive typical usage scenarios and environments in a broad range of geographies. Whenever possible, specifications are written in terms of survivable limits only. It is up to module developers to infer an appropriate qualification procedure that comports with good engineering practice and their own quality assurance methodology. The accompanying reference implementation sections provide test procedures, configurations, and other relevant information to demonstrate compliance to the specification.

8.1. Thermal Loads

It is critically important to restrict thermal loads from modules in the Ara architecture because of their effects on neighboring modules and the user. Since modules are nominally sealed from the environment, thermal loads generated are conducted either down into the endo frame and to the other modules or up into the module shell and to the user.

8.1.1. Thermal Loads - Specification

The Ara architecture has a thermal monitoring and management system. Modules must monitor their temperatures and adhere to temperature limits. Modules must provide measurements to the supervisory controller. When temperatures exceed predefined limits, modules may receive warnings, and in the worst case, the supervisory controller will cut power to the module.

8.1.2. Thermal Loads - Reference Implementation

Compliant temperature measurement components and associated logic will be provided in a future release. Details of compliant test procedures for thermal loads will also be provided in the future.

8.1.3. Thermal Loads - Prototype Specification

Prototype modules should limit their average power consumption to the recommended values in Table 8.1. The endo limits all modules to a maximum current draw of 2 A.

Table 8.1 - Prototype Module Thermal Loads

Module Type	Average Power
Application Processor	1.5 W
Generic Power Consumer	0.5 W
Charger	0.5 W
Power Storage	0.5 W

8.1.4. Thermal Loads - Prototype Reference Implementation

Tests to verify conformance to the specifications has not been implemented for the prototype.

8.2. Ambient Temperature and Humidity

8.2.1. Ambient Temperature and Humidity - Specification

Modules must be able to operate in ambient temperatures and humidities typical across a broad range of geographies. Stressing environments include combinations of high and low temperatures with high and low humidities. Modules must also operate through frequent and rapid extreme temperature and humidity changes such as when transitioning from a humid and hot tropical environment into an air-conditioned environment. Additionally, condensation can form during transitions between high and low humidity environments. Compliance tests must be designed to include such cases.

8.2.2. Ambient Temperature and Humidity - Reference Implementation

Details of compliant test procedures for ambient temperature and humidity will be provided in the future.

8.2.3. Ambient Temperature and Humidity - Prototype Specification

Prototype modules must operate in the range of ambient temperature and humidity limits defined in Table 8.2.

Table 8.2 - Prototype Ambient Temperature and Humidity Limits

	Temperature	Humidity
High	60 °C (140 °F)	95%RH
Low	-40 °C (-40 °F)	10%RH

8.2.4. Ambient Temperature and Humidity - Prototype Reference Implementation

Tests to verify conformance to the specifications have not been implemented for the prototype.

8.3. Electrostatic Discharge (ESD)

8.3.1. ESD - Specification

Ara modules must withstand electrostatic discharge thresholds congruent with the IEC 61000-4-2 standard. Modules must survive +/-8 kV contact discharges (at all sensitive locations such as the interface blocks and module base/shell interface) and +/- 15 kV, air discharge. Developers should use IEC-standard-compliant ESD generators.

8.3.2. ESD - Reference Implementation

IEC 61000-4-2 standard provides compliant test procedure for the ESD specification. Specific test points will be provided for each module type in the future along with functional tests performed before and after each discharge to verify proper module operation after each discharge.

8.3.3. ESD - Prototype Reference Implementation

Test procedures to verify conformance to the specifications has not been implemented for the prototype.

8.4. Shock

8.4.1. Shock - Specification

An Ara phone should be able to survive a 4 ft drop (TBD), typical of a user dropping the phone. The drop scenario can be implemented in multiple ways. Module level specifications is pending functional testing and will be provided in a future release.

8.4.2. Shock - Reference Implementation

Details of compliant test procedures for shock will be provided in the future. They will include examples of drop test configurations and test rigs, and functional tests performed after each drop to verify module operation.

8.4.3. Shock - Prototype Reference Implementation

Test procedures to verify conformance to the specifications has not been implemented in the prototype.

8.5. Vibration

8.5.1. Vibration - Specification

Modules must be able to survive typical vibration loads. A typical qualification for mobile devices involves 5 minutes of shaking in each orthogonal axis, in each direction at 30 Hz and a displacement of 0.06 in.

8.5.2. Vibration - Reference Implementation

Details of compliant test procedures for vibration will be provided in the future.

8.5.3. Vibration - Prototype Reference Implementation

Test procedures to verify conformance to the specifications has not been implemented for the prototype.

8.6. Electromagnetic Interference (EMI)

8.6.1. EMI - Specification

Modules must limit spurious and out-of-band emissions to limit interference to other modules in the phone. All modules must have a safety shield that acts as a Faraday cage on internal components mounted on the PCB. Additional specifications will be provided to verify EMI compliance in the future.

8.6.2. EMI - Reference Implementation

Details of compliant test procedures for EMI will be provided in the future.

Copyright © 2014 Google Inc. All rights reserved. Your use of this Ara Module Developers Kit (MDK) is expressly subject to the terms of the MDK License Agreement found at <http://projectara.com/mdk-license.txt>.

8.6.3. EMI - Prototype Reference Implementation

Test procedures to verify conformance to the specifications has not been implemented for the prototype.

8.7. Specific Absorption Rate (SAR)

8.7.1. SAR - Specification

RF-emitting modules must not exceed SAR limits, which are set by country-specific regulations. The Ara platform requires modules to meet SAR limits as required by law and carrier regulations.

8.7.2. SAR - Reference Implementation

Country-specific regulations typically specify detailed procedures and guidance required to demonstrate SAR compliance.

8.7.3. SAR - Prototype Reference Implementation

Test procedures to verify conformance to the specifications has not been implemented for the prototype.

9. Module Marketplace

This section provides guidance on submission, certification, and sale of Ara modules through the Ara Marketplace. Details will be provided in a future release.

10. Regulatory Certification

This section provides some guidance on legal, regulatory, and other certification requirements for Ara module developers. It is not intended to be definitive. Developers should consult official regulations to ensure compliance.

10.1. US Regulations

The following subsections provide some general guidance on radio frequency (RF), medical device, and carrier certification requirements in the US.

10.1.1. US RF Certification

The Federal Communication Commission (FCC) regulates RF-emitting devices in the US.

10.1.1.1. FCC Equipment Authorization - Specification

Under the Code of Federal Regulations (CFR), the FCC regulates all RF-emitting devices through a set of equipment authorization (EA) procedures designed to confirm compliance to the regulations before they can be imported into or marketed in the US. The FCC specifies several types of [EA procedures](#) that are congruent with device emissions and intended use. For low-powered digital devices that do not intentionally emit RF energy, the FCC requires a verification procedure that involves the manufacturer confirming devices operate within the appropriate technical standards. On the other hand, devices like cellular phones that utilize licensed radio frequencies and connect to the public telecommunications network require a more extensive certification procedure that includes testing to ensure RF emissions are below the safe threshold for specific absorption rate (SAR). The certification procedure also involves an official application filing with the FCC and associated fees.

10.1.1.2. FCC Equipment Authorization - Reference Implementation

[FCC equipment authorization processes for various module types will be provided in a future MDK release.](#)

10.1.1.2.1. FCC Equipment Authorization - Reference Implementation for Hobbyist/Maker
[Per CFR 47 Part 15.23, an equipment authorization is not required for devices that are not marketed and are built in quantities of five or less for personal use. While the exemption applies to Ara modules developed for personal use, developers should use good engineering judgement to ensure compliance with frequency and device safety guidelines including those specified in this document.](#)

10.1.1.2.2. FCC Equipment Authorization - Prototype/Experimental License
Per [CFR Part 5](#), an experimental license is required to utilize radio spectrum during experimentation, product development, and market trials. The majority of prototype development is not expected to involve radios; however, if a module developer requires an experimental license, they may obtain one directly from the FCC through the [experimental license system](#).

10.1.2. US Medical Device Certification

The US Food and Drug Administration (FDA) regulates medical devices in the US.

10.1.2.1. FDA Certification - Specification

Under the Code of Federal Regulations (CFR), the FDA regulates all medical devices in the US. Ara modules that are considered [medical devices](#) include devices intended for the diagnosis of disease or other conditions, or in the cure, mitigation, treatment, or prevention of disease, in man or other animals, or are intended to affect the structure or any function of the body of man or other animal. The FDA defines 3 classes of medical devices and associated certification procedures. Device classification depends on the intended use of the device and also upon indications for use. Classification for medical devices is determined by comparison with existing devices found in the FDA [classification database](#). Class 1 devices are exempt from Premarket Notification 510(k); most Class 2 devices require Premarket Notification 510(k); and most Class 3 devices require Premarket Approval. All medical devices have [general controls requirements](#) that cover various aspects of medical device development, marketing, sale, and post-market reporting.

10.1.2.2. Pulse Oximeter Module FDA Certification - Reference Implementation [None at this time.](#)

10.1.2.3. FDA Certification - Prototype Reference Implementation [None at this time.](#)

10.1.3. US Other Certification

10.1.3.1. US Carrier Certification TBD

10.2. International Certification

TBD