Open Source Sockets for the Internet of Things

Contents :

Table of Contents

Contents : 2
Introduction : 3
Overview : 4
Licensing : 5
thingSoC Compliant and Compatible : 5
thingSoC Socket Dimensions : 6
thingSoC Socket Connectors : 7
thingSoC Socket Stacking Levels : 8
thingSoC Socket RF Safe Zones : 10
thingSoC Socket Pin Configurations: 10
thingSoC Socket Signals : 11
thingSoC Socket Termination Resistors : 12
thingSoC Socket Interrupts : 12
thingSoC Compliant EEPROM Base : 13
thingSoC Firmware Meta Data Store : 15
thingSoC Auto-Discovery Algorithm : 16
thingSoC Application Programming Interface (API) : 17
Revision History : 18

Introduction :

There are already a number of popular and well-supported development platforms, like Arduino, Beaglebone, Raspberry Pi, and others, so why is thingSoC needed? While we work with and support all those platforms, each has a particular set of limitations, which thingSoC is designed to address.

There is currently a great deal of fragmentation and incompatibility in the market for development kits, tools, and products for the hobbyist and maker communities. While there is no real “one size fits all” grand solution to this, there are a number of common characteristics, goals, and interfaces that can make life much easier for cross-platform and cross-product interoperability.

One example of this fragmentation and incompatibility is evident with the Arduino platform, in it's various implementations. The Arduino is very popular, and a great number of third parties make “Arduino Shields”, as the Arduino expansion cards are known. However, some are 5.0 volt operation only, while others are 3.3V only, while still others use different pins for I2C, SPI and other control signals. It causes a great deal of confusion, and leads to customer frustrations with the incompatibilities and lack of interoperability.

Another example of the fragmentation is the proliferation of a number of incompatible “Gumstick” or breadboard friendly, low cost, microprocessor boards, that often do little more than simply bring out the microprocessor signals to the nearest pin. There are literally hundreds of these out in the market, with no common pin assignments, or the ability to utilize peripherals made for one flavor, on any other flavor of pin out. In our humble opinion, this trend is holding back a reusable solution to many of the recurring problems faced in embedded design work and rapid prototyping.

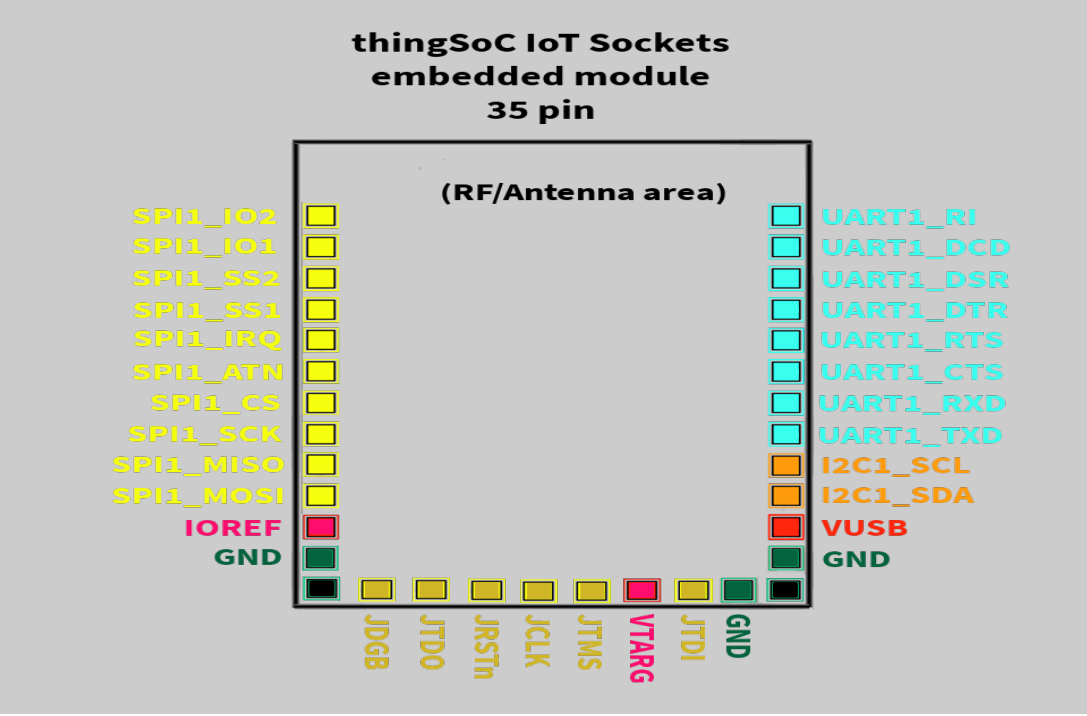
It was out of this frustration with incompatible products and development tools that the thingSoC project was conceived and developed. It has already been used in several Moxon Design and PatternAgents development programs, and is now being released as an Open Source Hardware/Software Specification.

Overview :

The thingSoC specification defines a physical, hardware socket system for inter-operable printed circuit boards, with a data centric firmware model for automatic device discovery, and a software API for interacting with the system.

The thingSoC sockets support both the basic low speed interface standards like I2C, SPI, and UART, but also support higher speed interfaces like USB 3.0, 10/100/1000, and PCI-E for faster performance IoT devices.

The following diagram shows the basic interfaces by group, and color-coded :



The thingSoC specification allows socket configurations from six (6) pins to thirty-five (35) pins for a single module. Multiple modules (double, triple, etc.) can be used to support a range of sizes, from small breakout boards to larger, high density input/output boards.

Licensing :

thingSoC Compliant and Compatible :

thingSoC Compliant devices implement at least the the Minimum Pin Configuration of six (6) pins, and the thingSoC EEPROM based auto-discovery mechanism in a thingSoC single, double, or triple sized PCB, with specified mounting holes, and display the thingSoC logo on the silkscreen.

thingSoC Compatible devices may fit and inter-operate with thingSoC based systems, but do not support the thingSoC EEPROM based auto-discovery mechanism or match the physical PCB size and specified mounting holes. This implies that **thingSoC Compatible** devices don't have access to thingSoC extended functions or display the thingSoC logo.

thingSoC Compliant devices implement a data centric firmware model for automatic device discovery which is compatible with the Linux 3.8 Kernal and Device Tree Overlays. The **thingSoC Compliant** meta-data store auto-discovery mechanism also implements additional features, however it is backward compatible with the Linux Beaglebone Cape Manager.

thingSoC Socket Dimensions :

thingSoC Compliant modules and sockets come in several standardized sizes with, and without specified mounting hole patterns. The Embedded Module format is assumed to be soldered (i.e. permanently mounted) to a host carrier board, where the stacking pin versions can include a mounting hole pattern for a secure, mechanical mounting in applications with vibration or other automotive level specifications.

Version 2.0 :

- A single thingSoC Embedded Module is :
  1.2 inches wide x 1.75 inches long.
- A dual thingSoC BB Embedded Module is :
  1.2 inches wide x 3.50 inches long.
-

- Multiple (3 to 16) thingSoC BB/EM modules can be placed on any size of larger board for more custom applications.

The default sizes have been chosen as a best compromise between cost, ease of assembly and ease of use. They have also been chosen to be compatible with a wide variety of existing packaging options.

thingSoC specification adopters may choose any size BASE or TEST carrier board size that meets their design or packaging requirements provided that the thingSoC sockets conform to the specified module sizes. Larger BASE or TEST carrier boards may provide a number of thingSoC sockets from one (1) to sixteen (16) per carrier board.

thingSoC Socket Connectors :

thingSoC compliant modules can have different I/O connector "bindings", in order to facilitate specific user requirements. The default thingSoC I/O connector binding is the “BB” or "Bread Board" type, which employs common 2.54mm pitch (0.1 inch) female header and male pin connections, and as the name suggests, is capable of being used with commonly available 2.54mm pitch breadboards and prototyping boards.

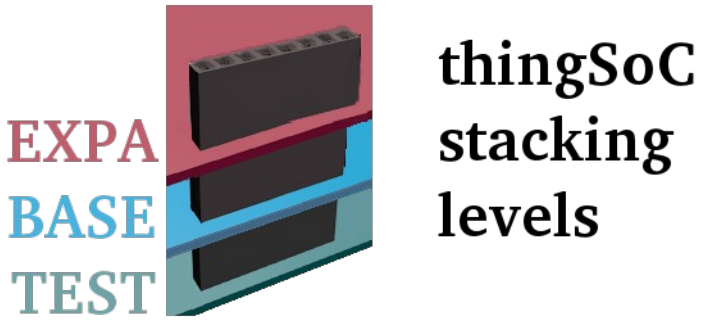
The thingSoc Embedded Module or “EM” format uses a Castellated I/O connector "binding" to provide low cost soldered-on module types that omit the cost, space and weight of connectors. They provide a permanent and inexpensive method to customize the I/O subsystems of many product classes.

Pin-Through-Hole (PTH) types, Surface Mount Device Top layer mounting types (SMD-TOP), and Surface Mount Device Bottom layer mounting types (SMD-BOT) are available in the thingSoC PCB libraries, giving thingSoC users a wide variety of connector choices. A Pogo-Pin ("PP" type) I/O connector type is also supported in the thingSoC libraries, in order to facilitate the creation of thingSoC "TEST" modules for production testing applications.

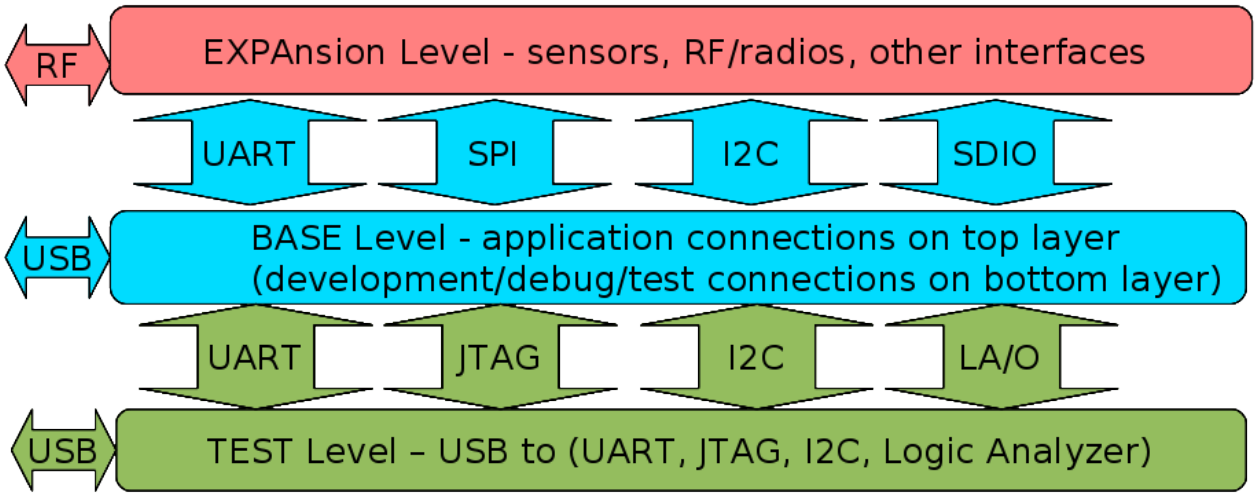
Other I/O connector type "bindings" are possible, and may be added to the thingSoC specification at some future date.

thingSoC Socket Stacking Levels :

thingSoC compliant module "stacking levels" refer to the type of board function, and it's vertical position in the physical stack of modules.



The following diagram shows the typical microprocessor I/O connection between the different stacking levels. TEST levels are typically used in development or production and quality assurance testing usage scenarios. They may also be used for field testing, data logging, and service quality testing usage scenarios.



- EXPAnsion Level(s)

  Any level above the "BASE" level is refered to be an "EXPA", or EXPAnsion level module. "EXPA" level modules use either "stack-thru" female header connectors, or surface mount (SMD) female headers on the top side of the board and surface mount (SMD) male pins on the bottom side of the board. The majority of thingSoC modules are "EXPA" level modules.
  A maximum of four (4) "EXPA" level modules can function in any thingSoC "BASE" socket (i.e. maximum stack height of four (4) "EXPA" level modules in any single thingSoC Socket).

- BASE Level

  There will be only one "BASE" level per thingSoC socket stack, and it generally contains the application processor and it's support subsystems. "BASE" level modules use either "stack-thru" female header connectors, or surface mount (SMD) female headers on the top side of the board and surface mount (SMD) male pins on the bottom side of the board.. Please note, this is very different than other small form factors systems, such as the Arduino, that treat the processor board (Uno, Leonardo, Due, etc.) as the bottom of the stack. Most thingSoC "BASE" level module implementations will utilize surface mount (SMD) female headers on the top side of the board, and leave the bottom connectors pads bare, (i.e. no male pins on bottom) in order to utilize Pogo-Pin ("PP") style test connectors on the "TEST" level.

- TEST Level

  Any level below the "BASE" level is refered to as an "TEST" level module. "TEST" level modules implement test, measurement, and monitoring functions for the thingSoC "BASE" socket, and any attached "EXPA" levels. The "TEST" level modules can include JTAG programming, debugging, and trace functionality. Oscilloscopes, Logic Analyzers, Protocol Analyzers, Emulators, and similar functions are implemented as "TEST" level modules in the thingSoC standard. "TEST" level modules eliminate common test lead/test jumper wiring and configuration problems by instrumenting standardized buses on known pins that can be easily probed and monitored.

thingSoC Socket RF Safe Zones :

There are designated "RF zones" in the thingSoC specification for supporting on-board antenna and RF interfaces. This is very important for creating a modular IoT platform that yields the maximum radio range with minimum RF interference.

The top 5.0 mm (0.2 inch) of PCB on a single size PCB, and the top and bottom 5.0 mm (0.2 inch) of PCB on double and triple size PCBS are designated "RF zones", and should have no Ground or Power fill (Flood or pour) of copper on any layer of the PCB. Thin traces are permitted for low speed signals, connectors, and indicator LEDs.

thingSoC Socket Pin Configurations:

thingSoC compliant modules can be from a minimum of six(6) pins to a maximum of eighty(80) pins per individual socket. The thingSoC module pin variations are known as "Pin Configurations", and support :

- 6 Pins - The Minimum Pin Configuration
  includes Vin, Vbat, 5V, Ground, I2C1_SDA, and I2C1_SCL,
  to form a bare minimum connection. The I2C bus is used for both configuration information and data exchange.
- 24 Pins - The Basic Pin Configuration
  adds UART and SPI bus pins for higher speed data exchange, as well as additional interrupt and control pins.
- 32 Pins - The Embeddable Module Configuration
  adds JTAG, and JTAG Pin multiplexing.
- 64 Pins - The Maximum Pin Configuration
  adds second I2C, second UART, Second SPI ports,
  as well as extended power, clock, and sleep control pins

thingSoC Socket Signals :

thingSoC signals are 3.3 volt (LVCMOS) levels, and may NOT necessarily be 5V tolerant. 5V tolerance is a recommended design practice only, not a requirement. Note that this is only the default pin configuration, other configurations are possible and defined by the thingSoC Firmware Meta Data Store (FMDS) EEPROM.

thingSoC modules can include the following signals in default configurations, note that the Direction listed is for a “host” socket; a “slave” or embedded module would have the signal Direction reversed :

Name	Direction	Function	Notes
GND	O	Power/Ground	Digital Ground
VUSB	O	Nominally 5.0 Volts	500mA Max
I2C_SDA	I/O	Primary I2C Bus Data	Connected to I2C EEPROM
I2C_SCL	I/O	Primary I2C Bus Clock	Connected to I2C EEPROM
UART1_TXD	O	UART Transmitted Data	
UART1_RXD	I	UART Received Data	
UART1_CTS	O	UART Clear to Send	
UART1_RTS	I	UART Request to Send	
UART1_DTR	I/A	UART Data Terminal Ready	
UART1_DSR	O/A	UART Data Set Ready	
UART1_DCD	I/A	UART Data Carrier Detect	
UART1_RI	I/A	UART Ring Indicator	
GND	O	Power/Ground	
IOREF	O	Nominally 3.3V	150mA Max
SPI_MOSI	O	SPI Master Out	
SPI_MISO	I	SPI Master In	
SPI_SCK	O	SPI Master Clock	
SPI_CS	O	SPI Chip Select	
SPI_ATN	O	SPI Attention/Reset	
SPI_IRQ	I	SPI Interrupt Request	
SPI_SS1	I/O/A	SPI Select 1	SS1 & SS2 allow for four (4)
SPI_SS2	I/O/A	SPI Select 2	SPI Slaves to be shared
SPI_IO1	I/O/A	SPI I/O 1	SS1 & SS2 allow for extra
SPI_IO2	I/O/A	SPI I/O 1	control lines for SPI devices
GND	O	Power/Ground	
VTARG	O	JTAG Progamming Voltage	(May be lower than IOREF)

Name	Direction	Function	Notes
JTDI	O	JTAG Test Data In	
JTMS	I/O	JTAG Test Mode	
JCLK	O	JTAG Test Clock	
JRST	O	JTAG Test Reset	
JTDO	I	JTAG Test Data Out	
JDBG	I	JTAG Debug Select	Tie low to enable debugging (used to multiplex JTAG pins)

Multiple Socket Naming Conventions :

thingSoC Compliant sockets are numbered from one (1) to sixteen (16), in a clockwise direction on the carrier board. The default naming is “PortSocket”, so UART1 is a UART signal on Socket #1, UART2 is a UART signal on Socket #2, and similarly for the SPI1, SPI2, I2C1, I2C2, etc.

Implementers will often utilize hubs and/or port multiplexors to support multiple thingSoC sockets, so Socket Zero (0) is reserved for the primary signals on the carrier board (i.e. I2C0, UART0, SPI0, etc). I2C bus switches that buffer the primary signals are recommended to reduce the capacitive loading and to facilitate different I/O voltage levels on IOREF.

thingSoC Socket Termination Resistors :

In order to insure proper operation, particularly during Shutdown and other low power modes, it is required to reference all pull-up, pull-down, other termination resistors be powered from the VBAT power rail. In order to minimize static current drain during Shutdown and other low power modes, it is recommended to provide a “power disable” to any unused pull-up, pull-down, other termination resistors.

The exception to this are the SLEEP, and WAKE signals,
which are active and powered during all sleep modes. They should be powered directly from powered from the VBAT power rail.

thingSoC Socket Interrupts :

All thingSoC socket interrupts are are required to utilize “active-low, level sensitive” behavior for external signals. Interrupts may be shared, and should be asserted until serviced.

thingSoC Compliant Data Store Memory:

thingSoC Compliant modules contain a memory/storage device containing data that identifies the board type, serial number, pin multiplexing, and other setup information. This allows thingSoC Compliant systems to automatically discover what boards have been installed into the system, and to configure itself properly to communicate with them.

thingSoC Compliant firmware is referred to as a "data driven architecture", because the memory/storage device contains only pure data, without any executable code that would be specific to a particular processor. We refer to the data structure contained in the memory/storage device as the thingSoC Firmware Meta Data Store (FMDS), and it contains information that can be used across different hardware platforms, to support a wide variety of different processor types. (i.e. low-power, high-speed, different architectures/vendors, etc.)

thingSoC Compliant modules implement the thingSoC Firmware Meta Data Store (FMDS) mechanism, which is compatible with the Linux 3.8 Kernal and Device Tree Overlays. The thingSoC Firmware Meta Data Store (FMDS) mechanism implements additional features, however it is backward compatible with the Linux/Beaglebone Black Cape Manager.

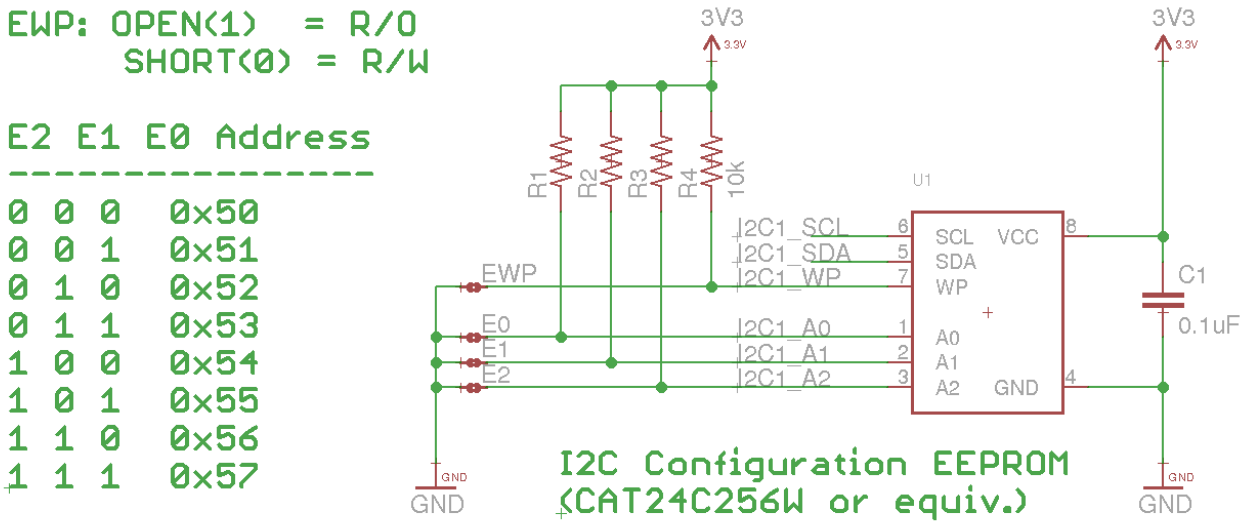
There are three (3) requirements for the memory device used for the thingSoC Firmware Meta Data Store (FMDS) :

- It be I2C bus compatible with 3.3V level signaling
- It be I2C bus addressable from addresses 0x50 to 0x57
- It appear as an I2C EEPROM register address model for data operations (i.e. CAT24C256 compatible)

So, any memory device that meets those requirements can be used, including an EEPROM emulator function, running within an I2C capable microprocessor.
If frequent data updates are required, the recommended practice is to use an SRAM, FRAM or NVSRAM in order to avoid data wear and leveling issues.

ThingSoC Example Data Store Memory (EEPROM) :

The following schematic diagram shows a thingSOC Compliant Data Store Memory, implemented using a common, low cost I2C EEPROM device :

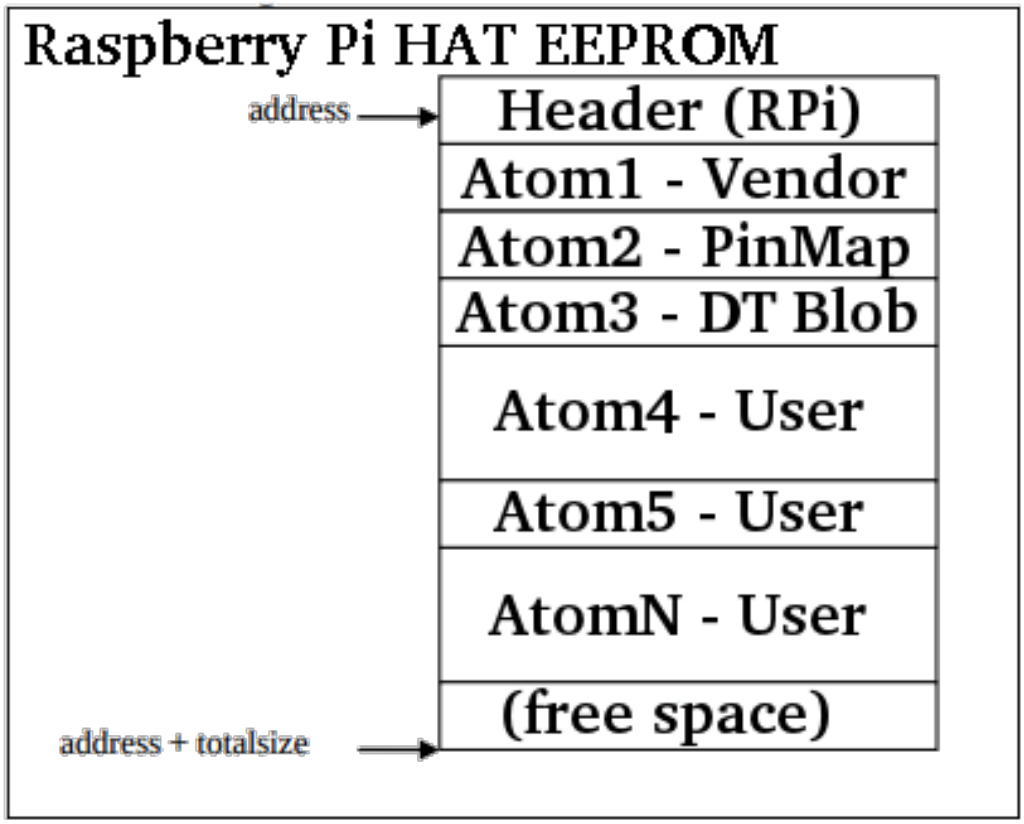


thingSoC Firmware Meta Data Store :

The thingSoC Firmware Meta Data Store (FMDS) mechanism can implement additional features by allowing additional "BLOBs" (Binary Level Objects) to be included in the memory/storage device, located immediately after the default FMDS "BLOB" in memory.

For example, an IPMI FRU (Intelligent Peripheral Management Interface - Field Replaceable Unit) "BLOB" could be located immediately after the default FMDS "BLOB", giving access IPMI format data as well. Each "BLOB" type can be identified by a unique "Magic Word" sequence at the beginning of the "BLOB" data section. This mechanism allows for complete backward compatibility, while allowing for the support of many new "BLOB" types within the thingSoC Firmware Meta Data Store (FMDS).

ThingSoC Socket FMDS “BLOB” Format :



thingSoC Auto-Discovery Algorithm :

(A simplified discovery algorithm in english)

The primary processor on the BASE level will use it's I2C1 Master port to :

- Read four (4) bytes from I2C Address 0x50, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the BASE Level
- Read four (4) bytes from I2C Address 0x51, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the TEST Level 1
- Read four (4) bytes from I2C Address 0x52, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the TEST Level 2
- Read four (4) bytes from I2C Address 0x53, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)

  If True, then a valid FMDS structure exists on the TEST Level 3
- Read four (4) bytes from I2C Address 0x54, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the EXPAnsion Level 1
- Read four (4) bytes from I2C Address 0x55, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the EXPAnsion Level 2
- Read four (4) bytes from I2C Address 0x56, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the EXPAnsion Level 3
- Read four (4) bytes from I2C Address 0x57, Offset 0x0000
  and compare to the Magic Word (0xAA, 0x55, 0x33, 0xEE)
- If True, then a valid FMDS structure exists on the EXPAnsion Level 4

thingSoC I/O Configuration Templates :

thingSoC Compliant modules can identify their required configuration via the thingSoC Firmware Meta Data Store (FMDS) mechanism. Each pin can be a member of an I/O group, that allows for a simple auto-configuration method.

ThingSoC Pin Groups :

There are sixty-four pin groups (6 bits) as follows :

0U No Group – Default
1U I2C Group
2U UART Group
3U SPI Group
4U Analog Group
5U JTAG Group
6U SD/MMC Group
7U Ethernet/Phy Group
8U CAN Group
9U
…
63U Reserved

This mechanism allows for more automatic configuration and the ability to recognize incompatible configurations of attached peripherals.

thingSoC Application Programming Interface (API) :

thingSoC Compliant systems are compatible with RSVP-SIS, the Reference System Virtual Platform - Software Interface Standard, an open source, vendor agnostic, application programming interface that has ports to several different IDE's.

Please see the RSVP-SIS Organization Website for more information : http://www.rsvpsis.com

Revision History :

04-01-2015 : First Preliminary Draft – released for review

04-13-2016 : Version 2.0