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.

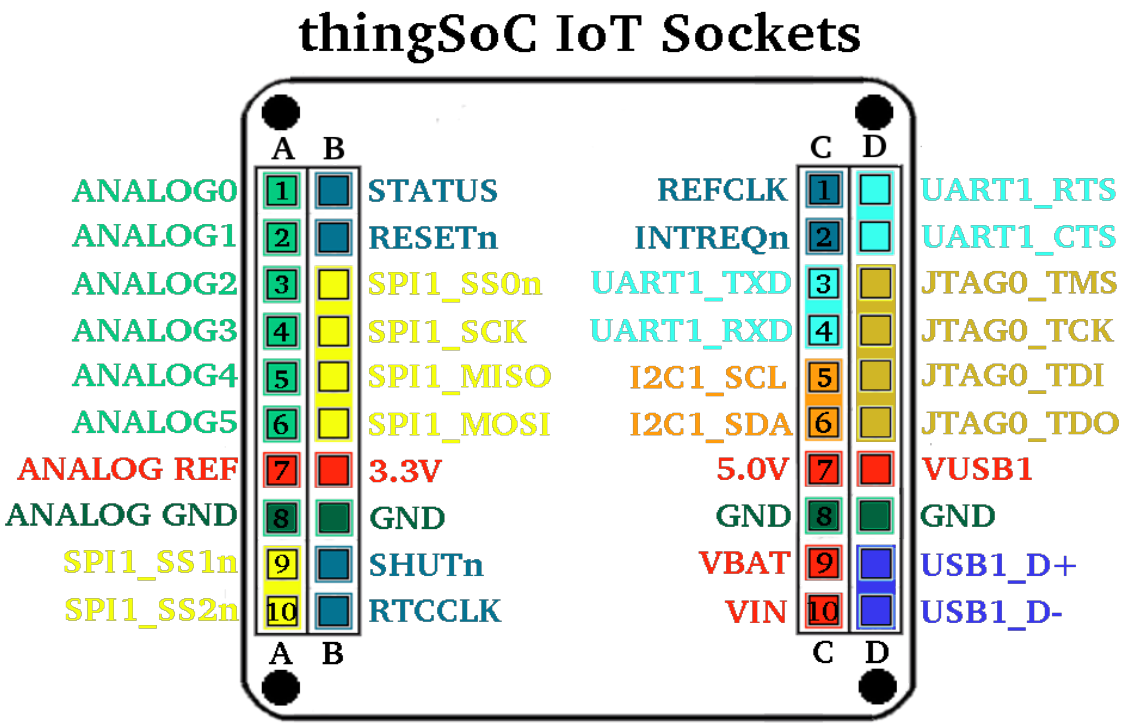
It was out of this frustration with incompatible products and development tools that the thingSoC project was conceived and developed. It has already been used in several Moxon Design and PatternAgents development programs, and is now being released as an Open Source Hardware/Software Specification.

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) to eighty (80) pins, supporting a range 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.

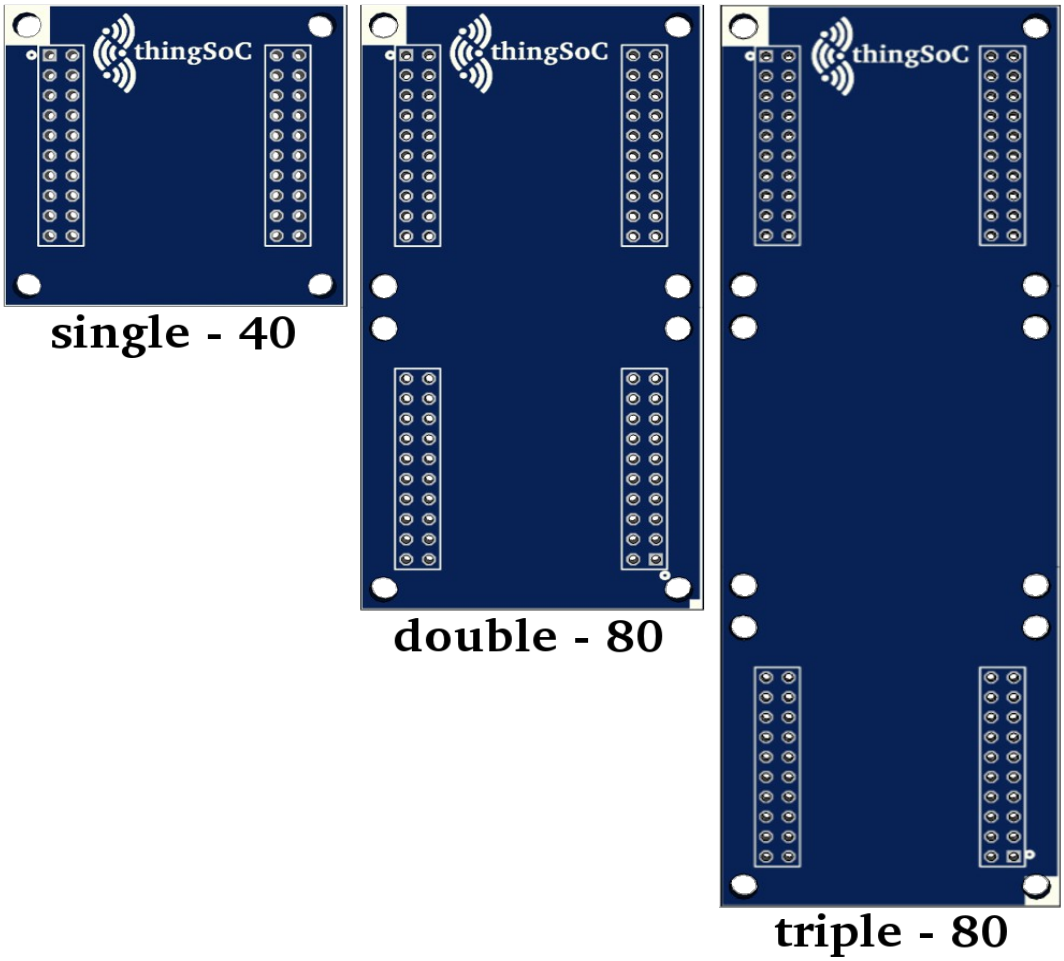
thingSoC Compatible devices may fit and inter-operate with thingSoC based systems, but do not support the thingSoC EEPROM based auto-discovery mechanism or match the physical PCB size and specified mounting holes. This implies that **thingSoC Compatible** devices don't have access to thingSoC extended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

- A single thingSoC module is 38.1mm long by 38.1mm wide (1.5 inches x 1.5 inches).
- A double thingSoC module is 79.2mm long by 38.1mm wide (3.12 inches x 1.5 inches).
- A triple thingSoC module is 114.8mm long by 38.1mm wide (4.52 inches x 1.5 inches).



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.

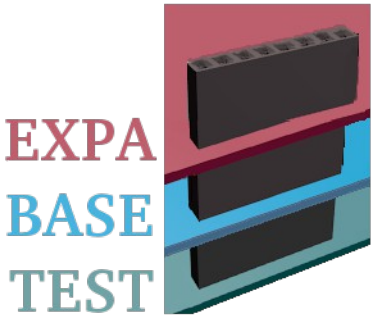
Pin-Through-Hole (PTH) types, Surface Mount Device Top layer mounting types (SMD-TOP), and Surface Mount Device Bottom layer mounting types (SMD-BOT) are available in the thingSoC PCB libraries, giving thingSoC users a wide variety of connector choices. A Pogo-Pin ("PP" type) I/O connector type is also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

A “CAS” or Castellated I/O connector "binding" is also in development for low cost soldered-on module types that omit the cost, space and weight of connectors.

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

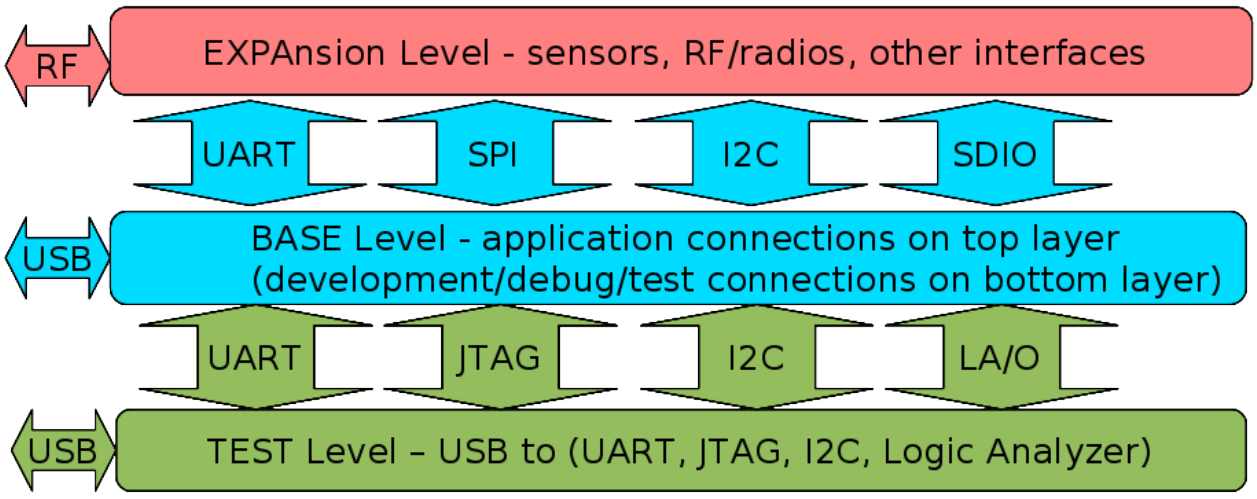
thingSoC Socket Stacking Levels :

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



thingSoC stacking levels

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.
- 20 Pins - The Basic Pin Configuration
  adds UART and SPI bus pins for higher speed data exchange, as well as additional clocks, interrupt and power saving shutdown control pins.
- 40 Pins - The Standard Pin Configuration
  adds USB2.0, JTAG, and Analog.
- 80 Pins - The Maximum Pin Configuration
- adds Superspeed USB3.0, PCI-E, or 10/100 Ethernet, second UART, Second SPI ports.

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.

thingSoC modules can include the following signals in default configurations :

Pin	Name	Type	Function
1A	ANALOG0	Analog I/O	Analog Input/Output. Typically an Input to the BASE level, driven from a sensor on an EXPAnsion level.
2A	ANALOG1	Analog I/O	Analog Input/Output. Typically an Input to the BASE level, driven from a sensor on an EXPAnsion level.
3A	ANALOG2	Analog I/O	Analog Input/Output. Typically an Input to the BASE level, driven from a sensor on an EXPAnsion level.
4A	ANALOG3	Analog I/O	Analog Input/Output. Typically an Input to the BASE level, driven from a sensor on an EXPAnsion level.
5A	ANALOG4	Analog I/O	Analog Input/Output. Typically an Input to the BASE level, driven from a sensor on an EXPAnsion level.
6A	ANALOG5	Analog I/O	Analog Input/Output. Typically an Input to the BASE level, driven from a sensor on an EXPAnsion level.
7A	AREF	Power	Analog Voltage Reference and Supply. Typically 1.8 volts to 3.3 volts, maximum 50mA source from BASE level.
8A	ANALOG GROUND	Power	Analog Ground.
9A	SPI1_SS1n (A6)	Analog Output	Analog or Digital Input/Output. Typically an Output from the BASE level, used an extra SPI1 device selection.
10A	SPI1_SS2n (A7)	Analog Output	Analog or Digital Input/Output. Typically an Output from the BASE level, used an extra SPI1 device selection.
1B	STATUS	Analog Input	Analog or Digital Input/Output. Typically an input to the BASE level, used as a socket status indicator.
2B	RESETn	Open Collector	Shared I/O line, active low. Any "watchdog" can pull this line low to trigger a hardware reset operation.
3B	SPI1_SS0n	GPIO	Typically an Output from the BASE level, used as the primary SPI1 device selection. Disabled during Shutdown.
4B	SPI1_SCK	GPIO	Typically an Output from the BASE level, used as the primary SPI1 clock. Disabled during Shutdown.
5B	SPI1_MISO	GPIO	Typically an Input to the BASE level, used as the primary SPI1 data input. Disabled during Shutdown.
6B	SPI1_MOSI	GPIO	Typically an Output from the BASE level, used as the primary SPI1 data output. Disabled during Shutdown.
7B	3.3V	Power	Digital Supply. Maximum 250mA @3.3 volts nominal. Voltage range is 3.2v – 3.5v. Disabled during Shutdown.
8B	DIGITAL GROUND	Power	Digital Ground
9B	SHUTn	Open Collector	Shared I/O line, active low. Any "watchdog" can pull this line low to trigger a hardware Shutdown operation.
10B	RTCCLK	GPIO	Typically an Output from the BASE level, used as the primary 32,768 Khz Real-Time Clock. Active during Shutdown.
1C	REFCLK	GPIO	Typically an Output from the BASE level, used as the primary 8-25Mhz Reference Clock. Disabled during Shutdown.
2C	INT_REQn	Open Collector	Shared I/O line, active low. Used to request an I2C1 interrupt operation, or a Wake-from-Sleep operation.
3C	UART1_TXD	GPIO	Typically an Output from the BASE level, used as the primary UART1 data output. Disabled during Shutdown.
4C	UART1_RXD	GPIO	Typically an Input to the BASE level, used as the primary UART1 data input. Disabled during Shutdown.
5C	I2C1_SCL	I2C	Shared I/O line, used as the primary I2C1 clock line. Disabled during Shutdown.
6C	I2C1_SDA	I2C	Shared I/O line, used as the primary I2C1 data line. Disabled during Shutdown.
7C	5.0V	Power	Digital Supply. Maximum 250mA @5.0 volts nominal. Voltage range is 4.2v – 5.5v. Disabled during Shutdown.
8C	DIGITAL GROUND	Power	Digital Ground
9C	VBAT	Power	Battery Input, typically wired directly to a single cell Li-Po battery. Min. 500mA, Max 2A. Always Active.
10C	VIN	Power	External High Voltage Supply Input. Min. 500mA, Max 2A. Voltage range 5.5V to 28V max. Disabled during Shutdown.
1D	UART1_RTS	GPIO	Typically an Output from the BASE level, used as UART1 Request-to-Send. Disabled during Shutdown.
2D	UART1_CTS	GPIO	Typically an input to the BASE level, used as UART1 Clear-to-Send. Disabled during Shutdown.
2D	JTAG0_TMS	GPIO	Typically an input to the BASE level, used as JTAG Test-Mode-Select. Disabled during Shutdown.
4D	JTAG0_TCK	GPIO	Typically an input to the BASE level, used as JTAG Test-Clock. Disabled during Shutdown.
5D	JTAG0_TDI	GPIO	Typically an output from the BASE level, used as JTAG Test-Data-Input. Disabled during Shutdown.
6D	JTAG0_TDO	GPIO	Typically an input to the BASE level, used as JTAG Test-Data-Output. Disabled during Shutdown.
7D	USB1_VBUS	Power	USB1 Digital Supply. Maximum 850mA @5.0 volts nominal. Voltage range is 4.2v – 5.5v. Disabled during Shutdown.
8D	DIGITAL GROUND	Power	Digital Ground
9D	USB1_DP	USB	USB1 Data Plus Signal line.Disabled during Shutdown and/or USB Disconnect. Used with BASE level USN Hubs.
10D	USB1_DM	USB	USB1 Data Minus Signal line.Disabled during Shutdown and/or USB Disconnect. Used with BASE level USN Hubs.

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

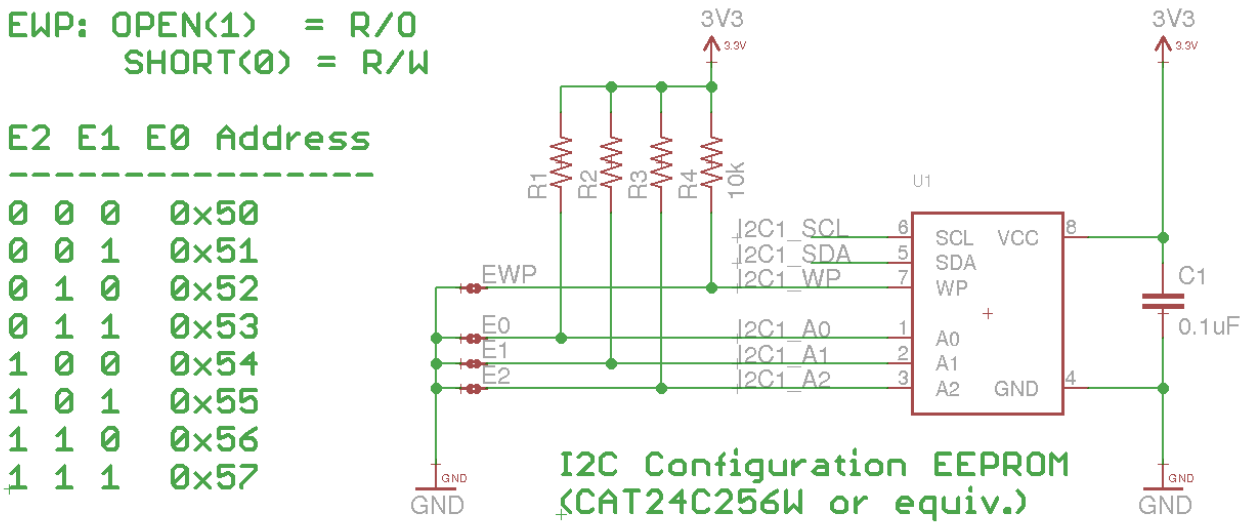
There are three (3) requirements for the memory device used for the thingSoC Firmware Meta Data Store (FDMS) :

- 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 (FDMS) 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 FDMS "BLOB" in memory.

For example, an IPMI FRU (Intelligent Peripheral Management Interface - Field Replaceable Unit) "BLOB" could be located immediately after the default FDMS "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 (FDMS).

ThingSoC Socket FMDS “BLOB” Format :

Name	Offset	Size	Format	Contents
Header	0	4	BIN	0xAA, 0x55, 0x33, 0xEE
EEPROM Revision	4	2	ASCII	EEPROM Revision number (e.g. “A1”)
Board Name	6	32	ASCII	Board Name : up to developer of the board as to what they call the board.
Version	38	4	ASCII	Hardware version code : up to the developer to determine their version format.
Manufacturer	42	16	ASCII	Manufacturer Name : corporate or individual’s name.
Part Number	58	16	ASCII	Part number : up to the developer to determine their part numbering scheme.
Number of Pins	74	2	HEX	Number of pins used by the daughter board (including power pins). This is HEX value of the number of pins, in decimal.
Serial Number	76	12	ASCII	Serial number of the board. This is a 12 character string which is: WWYY&&&&nnnn Where: WW = 2 digit week of the year of production Serial Number YY = 2 digit year of production &&&&=Assembly number or product. nnnn = incrementing board number for that week of production
Pin Usage Array	88	148	BIN	Two bytes for each configurable pin on the thingSoC Socket MSB LSB Bit order: 15 14 .....1..0 Bit 15.....Pin is used or not.....0=Unused by cape 1=Used by Socket Bit 14-13.....Pin Direction.....1 0=Output 01=Input 11=BDIR Bits 12-7.....Reserved.....should be all zeros Bit 6.....Slew Rate .....0=Fast 1=Slow Bit 5.....Rx Enable.....0=Disabled 1=Enabled Bit 4.....Pull Up/Dn Select.....0=Pulldown 1=PullUp Bit 3.....Pull Up/DN enabled.....0=Enabled 1=Disabled Bits 2-0 .....Mux Mode Selection.....Mode 0-7
VDD_3V3 Current Sink	236	2	BIN	Maximum current in mA, this is HEX value of the current in decimal 1500mA=0x05 0xDC 325mA=0x01 0x45
VDD_5V0 Current Sink	238	2	BIN	Maximum current in mA, this is HEX value of the current in decimal 1500mA=0x05 0xDC 325mA=0x01 0x45
SYS_5V0 Current Sink	240	2	BIN	Maximum current in mA, this is HEX value of the current in decimal (e.g. 1500mA=0x05 0xDC, 325mA=0x01 0x45)
VDD_5V0 Current Src	242	2	BIN	Is the board supplying power on the main VDD_5V 00 00 = No Current Sink Only 1-0xFFFF current supplied in mA (e.g. 1500mA=0x05 0xDC, 325mA=0x01 0x45)
VBAT_Current_Sink	244	2	BIN	Maximum current in mA, this is HEX value of the current in decimal (e.g. 1500mA=0x05 0xDC, 325mA=0x01 0x45)
VIN_Current_Sink	246	2	BIN	Maximum current in mA, this is HEX value of the current in decimal (e.g. 1500mA=0x05 0xDC, 325mA=0x01 0x45)
Reserved	248	2	BIN	Reserved
Reserved	250	2	BIN	Reserved
Reserved	252	2	BIN	Reserved
Checksum	254	2	BIN	Checksum (16bit) of this section of the EEPROM

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 FDMS 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 FDMS 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 FDMS 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 FDMS 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 FDMS 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 FDMS 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 FDMS 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 FDMS structure exists on the EXPAnsion Level 4

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

@TODO : Add “BB” stack-thru connector height & clearance parameters.