UP Core Carrier Board EEPROM specification rev. 1.0

Abstract:

This specification defines a common hardware detection and configuration scheme for the UP Core carrier boards.

Revision History

Revision	Description	Date	Author
1.0	Finalised document for public release	20/12/2017	Daniele Cleri
0.4	Added documentation about GMAP and SSDT Custom Manufacturer data as agreed with AAEON Bios development team. Updated ASL code example	13/12/2017	Nicola Lunghi
0.3	Restricted EEPROM Address Range (0x50-0x53) Updated ASL code example Various format corrections	25/09/2017	Nicola Lunghi
0.2	Revision based on feedback, minor tweaks	20/07/2017	Javier Arteaga

TABLE OF CONTENTS

1 INTRODUCTION

The purpose of this document is to depict the scheme used by the UP Core main board to automatically detect and configure carrier boards from firmware. This is achieved by reading an EEPROM connected to the Cherry Trail SoC via two I2C pins on the 100-pin connector (exHAT). The properties and contents of this EEPROM are defined below.

2 REFERENCES

1. Raspberry Pi EEPROM image format spec, 17-07-2017
2. ACPI DSD properties documentation from the Linux kernel, 17-07-2017
3. B+ ADD-ON BOARD / HAT DESIGN GUIDE - EEPROM specification

3 HARDWARE ASSUMPTIONS

1. The 100-pin exHAT will bring out I2C0 from the Cherry Trail SoC on these pins:
   - I2C0 SDA: pin 73
   - I2C0 SCL: pin 75
2. Any carrier board that supports automatic board detection will provide an EEPROM connected to the I2C pins listed at point 1.
3. EEPROM size is 4 kilobyte (**32 kilobit**), so it can contain any SSDT blob.
4. The EEPROM address can be overridden by the user via hardware switches or jumpers, to resolve address collisions. The available range is **0x50**-**0x53**.
5. For ease of use, the default EEPROM address should be different for different board types. As there are few addresses, a default address should be shared between two carrier boards only when they are thought to be incompatible or unlikely to be used in conjunction.

For other EEPROM hardware related specification please refer to the document at point 3 of the references.

4 EEPROM IMAGE FORMAT

4.1 Overall Structure

Field	Content
HEADER	EEPROM Header
ATOM1	Vendor info atom (**Required**)
ATOM2	GPIO map atom (Optional, only required for RPi shield compatibility)
ATOM3	DT blob atom (Optional, only required for RPi HAT spec compliance)
ATOM4	Manufacturer Custom Data

Table 1 Overall Structure

4.2 Header Structure

Bytes	Field	Notes
4	signature	0x52 0x2D 0x50 0x69 ("R-Pi" in ASCII)
1	version	EEPROM format version: 0x01
1	reserved	0x00
2	numatoms	Number of atoms contained in the EEPROM (ex 2)
4	eeplen	1

Table 2: Header Structure

4.3 Atom Structure

Bytes	Field	Notes
2	type	Atom type (see below)
2	count	Incrementing atom count
4	dlen	Length in bytes of data + CRC
N	data	(dlen – 2) bytes of atom data (see below)
2	crc16	CRC-16-CCITT of entire atom (type, count, dlen, data)

Table 3: Atom Structure

4.4 Atom Types

Hex Code	Description
0x0000	invalid
0x0001	Vendor info
0x0002	GPIO map
0x0003	Linux Device Tree blob
0x0004	Manufacturer Custom Data

Table 4 Atom Types

4.5 "Vendor Info" Atom (type 0x0001)

Bytes	Field	Notes
16	uuid	It must be a valid UUID and it must be different for different carrier board types, at least. Optionally, it can also be unique per every carrier board unit made.
2	pid	Product ID (see Product ID table)
2	pver	Board Revision
1	vslen	Vendor string length (bytes)
1	pslen	Product string length (bytes)
X	vstr	ASCII vendor string e.g. "AAEON"
Y	pstr	ASCII product string e.g. "UP-CRST02 rev A"

Table 5: "Vendor Info" Atom structure

4.5.1 PID values used for UP Core Carrier Boards

Product ID	Carrier Board
1000	UP-CREX
1001	UP-CRST01
1002	UP-CRST02

Table 6: PID values used for UP Core Carrier Boards

4.6 "GPIO Map" Atom (type 0x0002)

To be used for Raspberry Pi format compatibility only.

This atom should be used exclusively to create HATs compatible with both Raspberry Pi and UP (see References). The UP Board BIOS should ignore the content of this atom.

4.7 "Linux Device Tree blob" Atom (type 0x0003)

To be used for Raspberry Pi format compatibility only.

This atom should be used exclusively to create HAT/Carrier Boards compatible with both Raspberry Pi and UP (see References). The UP Board BIOS should ignore the content of this atom.

4.8 "Manufacturer Custom Data" Atom (type 0x0004)

This atom can be used to store any custom data defined by the HAT/Carrier Board manufacturer or to allow the BIOS to interpret the content of this field as an ACPI table or an FPGA/GPIO pin initial configuration.

To do so, the first 4 bytes of this field must be initialised as follow:

First 4 byte of binary data (hex)	ASCII Value	Expected Data Type
0x53 0x53 0x44 0x54	SSDT	Compiled version of the ACPI table of peripherals on the hat
0x47 0x4D 0x41 0x50	GMAP	initial configuration of the fpga gpio pin

Table 7: Custom Manufacturer Data identification bytes

SSDT:

If the first 4 bytes of the field are set to start with this value, the entire content of the field will be interpreted as a compiled version of the ACPI fragment for the peripheral contained on the carrier board as obtained by compiling ACPI tables with iasl compiler tool.

GMAP

If the first 4 bytes of the field are set to start with this value, the following 22 bytes will be interpreted as each pin of the pin controller.

This configuration should be used only if the board is equipped with a pinctrl device and it should also configure the SOC Gpio function. Moreover, in case the board is carrying an FPGA with the new protocol, it should be configured the direction of the corresponding pin in the FPGA.

All GPIO pins are set by default with the value of “multifunction”.

Each byte has the following meaning:

Byte hex value	Byte Binary Value	Data Type
0x00	0000 0000	Don’t care: don’t set this gpio, configure depending on other pins
0x02	0000 0010	Pin set as GPIO IN
0x03	0000 0011	Pin set as GPIO OUT
0x04	0000 0100	Pin set as multi-function -> I2C, SPI, UART, I2S depending on the pin

Table 8 GPIO bytes meaning

5 UEFI Firmware Boot Duties

For OS portability, UEFI firmware **must** perform the carrier board hardware detection.

This means that the support for I2C reads through EFI code is necessary.

1. Scan the I2C0 bus, all addresses 0x50 to 0x53.
2. Detect if the EEPROM contain a valid HEADER as described in the Header Structure Specification
3. Display in the BIOS menu the information contained in the Vendor Info Atom (type 0x0001)
    1. ASCII Vendor string
    2. ASCII Product StringProduct ID
    3. Board Revision
4. Look for a Custom Manufacturer data atom (type: 0x0004, first data bytes: “SSDT”)

1. If the atom is found, load its data as an SSDT table
2. *An optional Custom Manufacturer data atom (type: 0x0004, first data bytes: "GMAP") could be included to specify the initial configuration of each pin of a GPIO controller if present. Please refer to the* "GPIO Map" Atom (type 0x0002) *of this document for more information.*

Notes:

- This process should happen as early as possible in the boot process.
- Failure to process EEPROM data for an I2C address **must not** stop the scanning process on other I2C addresses.
- Custom Manufacturer data is completely optional so the BIOS should work correctly also if this data is missing / invalid.
- Some carrier board configurations could conflict with each other. The firmware **must** detect these conflicts and must not attempt to set invalid configurations (for example, two carrier boards driving the same GPIO pin to a different state, malformed GMAP data).

6 Example: ACPI SSDT for a dummy carrier board

Please note: this is a draft, sample SSDT for illustrative purposes only and could be subjected to changes.

```
DefinitionBlock ("", "SSDT", 1, "AAEON", "UPCR01", 1)
{
  // References to GPIO controller devices from Cherry Trail
  External (_SB_.GPO0, DeviceObj)
  External (_SB_.GPO1, DeviceObj)
  External (_SB_.GPO2, DeviceObj)
  External (_SB_.GPO3, DeviceObj)
  Scope (\_SB)
  {
    // any ACPI properties or methods that apply to the board go here...
    {
    Name (FCFG, Zero)
    Name (RCFG, Zero)
    // _HID: Hardware ID:
    //      AANT0F00: UP1
    //      AANT0F01: UP^2
    //      AANT0F02: UP core CREX
    //      AANT0F03: UP core CRST02
    Name (_HID, "AANT0F02")
    // _DDN: DOS Device Name
    Name (_DDN, "UP-CREX rev A0.3_0_0")
    Name (_UID, One)  // _UID: Unique ID
    // _HRV: Hardware Revision:
    //       1 for FPGA protocol 1 (first UP board)
    //       2 for FPGA protocol 2
    Name (_HRV, 2)
    // _DEP: Dependencies
    Name (_DEP, Package (0x04)
    {
      \_SB.GPO0,
      \_SB.GPO1,
      \_SB.GPO2,
      \_SB.GPO3,
    })
    Name (_CRS, ResourceTemplate ()
    {
      /*
      * CPLD SoC GPIO Pins 0-27
```

```
* These GPIOs are ordered relative to the corresponding
* bit position in the CPLD pin direction "DIR" register
*/
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 0
"\\_SB.GPO0", 0) {61}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 1
"\\_SB.GPO0", 0) {65}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 2
"\\_SB.GPO0", 0) {60}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 3
"\\_SB.GPO0", 0) {63}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 4
"\\_SB.GPO3", 0) {7}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 5
"\\_SB.GPO3", 0) {79}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 6
"\\_SB.GPO3", 0) {6}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 7
"\\_SB.GPO3", 0) {3}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 8
"\\_SB.GPO3", 0) {4}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 9
"\\_SB.GPO0", 0) {20}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 10
"\\_SB.GPO0", 0) {16}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 11
"\\_SB.GPO3", 0) {5}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 12
"\\_SB.GPO3", 0) {1}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 13
"\\_SB.GPO0", 0) {96}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 14
"\\_SB.GPO0", 0) {92}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 15
"\\_SB.GPO0", 0) {94}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 16
"\\_SB.GPO0", 0) {97}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 17
"\\_SB.GPO2", 0) {21}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 18
"\\_SB.GPO2", 0) {24}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 19
"\\_SB.GPO2", 0) {15}
GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 20
```

```
    "\\_SB.GPO2", 0) {22}
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone, // 21
    "\\_SB.GPO2", 0) {20}

    // CPLD Configuration Interface Pins
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone,
    "\\_SB.GPO0", 0) {78} // CPLD OE
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone,
    "\\_SB.GPO2", 0) {26} // CPLD STROBE
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone,
    "\\_SB.GPO2", 0) {16} // CPLD DATAIN
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone,
    "\\_SB.GPO2", 0) {18} // CPLD CLEAR
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone,
    "\\_SB.GPO1", 0) {23} // CPLD RESET
    GpioIo (Exclusive, PullDefault, 0, 0, IoRestrictionNone,
    "\\_SB.GPO2", 0) {17} // CPLD DATAOUT
  })

  // ACPI DSD property names used by the OS
  Name (_DSD, Package ()
  {
    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
    Package ()
    {
      // 40-pin header pins
      Package () { "external-gpios", Package() {
      ^PCTL,  0, 0, 0,
      ^PCTL,  1, 0, 0,
      ^PCTL,  2, 0, 0,
      ^PCTL,  3, 0, 0,
      ^PCTL,  4, 0, 0,
      ^PCTL,  5, 0, 0,
      ^PCTL,  6, 0, 0,
      ^PCTL,  7, 0, 0,
      ^PCTL,  8, 0, 0,
      ^PCTL,  9, 0, 0,
      ^PCTL, 10, 0, 0,
      ^PCTL, 11, 0, 0,
      ^PCTL, 12, 0, 0,
      ^PCTL, 13, 0, 0,
      ^PCTL, 14, 0, 0,
      ^PCTL, 15, 0, 0,
      ^PCTL, 16, 0, 0,
```

```
        ^PCTL, 17, 0, 0,
        ^PCTL, 18, 0, 0,
        ^PCTL, 19, 0, 0,
        ^PCTL, 20, 0, 0,
        ^PCTL, 21, 0, 0,
        }},
        // CPLD communication line pins by name
        Package () { "enable-gpio",  Package() { ^PCTL, 22, 0, 0 }},
        Package () { "strobe-gpio",  Package() { ^PCTL, 23, 0, 0 }},
        Package () { "datain-gpio",  Package() { ^PCTL, 24, 0, 0 }},
        Package () { "clear-gpio",   Package() { ^PCTL, 25, 0, 0 }},
        Package () { "reset-gpio",   Package() { ^PCTL, 26, 0, 0 }},
        Package () { "dataout-gpio", Package() { ^PCTL, 27, 0, 0 }},
      }
    })
    }
  }
```