Raspberry Pi ADD-ONs habitat

Compatibility Specification

Revision Log:

Date	Author	Reason for update
14-Mar-2014	Ivan Zilic	Initial Draft

1 Scope

The scope of this document is to specify Raspberry Pi add-ons requirements in terms of:

- Hardware detection
- Unified software support
- Conflicts and malfunction prevention

Covered interfaces are GPIO port and USB.

2 Entities

The specification is based on the following main features for entities:

- Add-on board type has unique ID, which is a combination of Vendor ID and Product ID,
- Global database contains relations between the add-on boards ID and properties,
- Raspberry Pi can check boards coexistence issues,
- Application (Scratch) can find out what are capabilities of attached add-on board(s) and can communicate with them in a generalized way.

3 GPIO add-on boards

In order to detect present boards and to query them in a generalized way, one dedicated GPIO pin is required. GPIO pin 25 is reserved exclusively and is not to be used for any other purpose by any attached add-on board.

3.1 Boards configurations

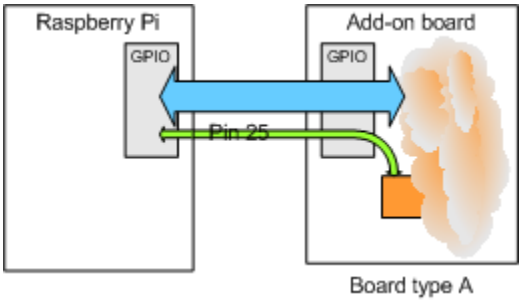
This specification allows the following combinations of compatible add-on boards:

- Single add-on board,
- Multiple add-on boards stacked:
  - Each board is unique,
  - Several equal boards present alone,
  - Several equal boards present next to one or more unique boards.

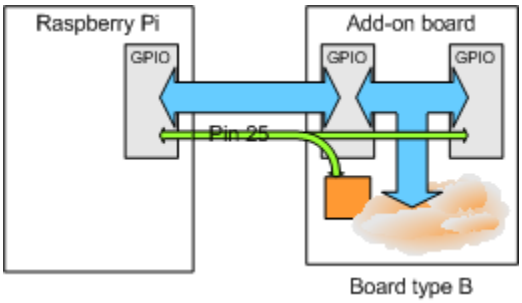
3.1.1 Single add-on board

Any compatible add-on board can be present alone.

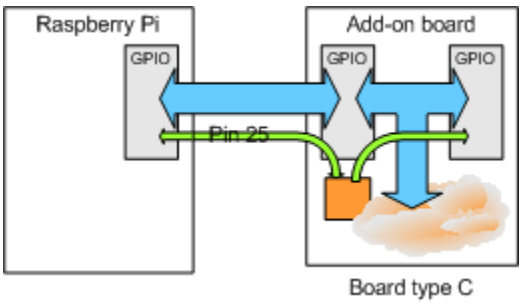
The board type A doesn't have male GPIO connector suitable tor stacking:



The board type B has male GPIO connector and is suitable for stacking. GPIO pin 25 is directly routed to male GPIO connector (one unique board supported):



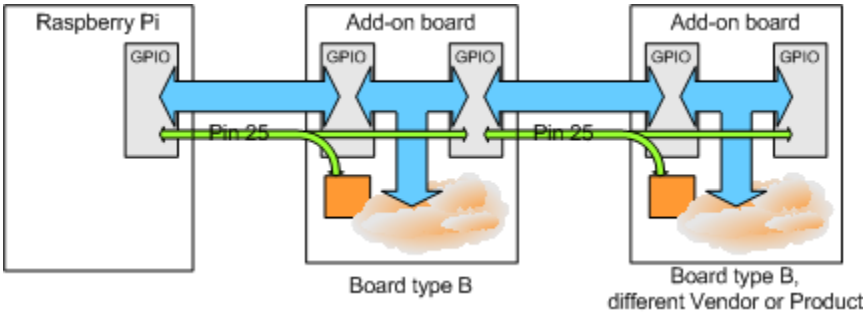
The board type C has male GPIO connector and is suitable for stacking. GPIO pin 25 is daisy-chained to male GPIO connector (multiple equal boards supported):



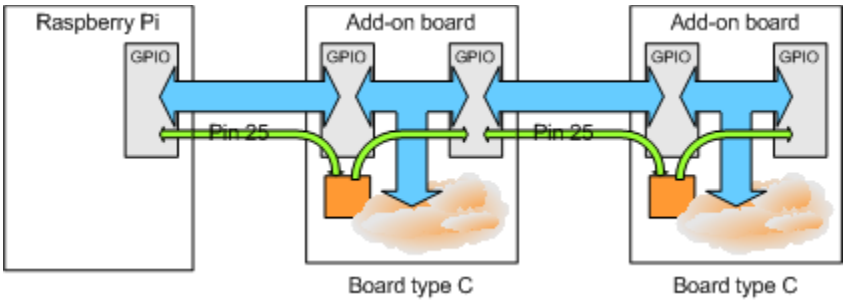
3.1.2 Multiple add-on boards

Boards can be stacked in several configurations.

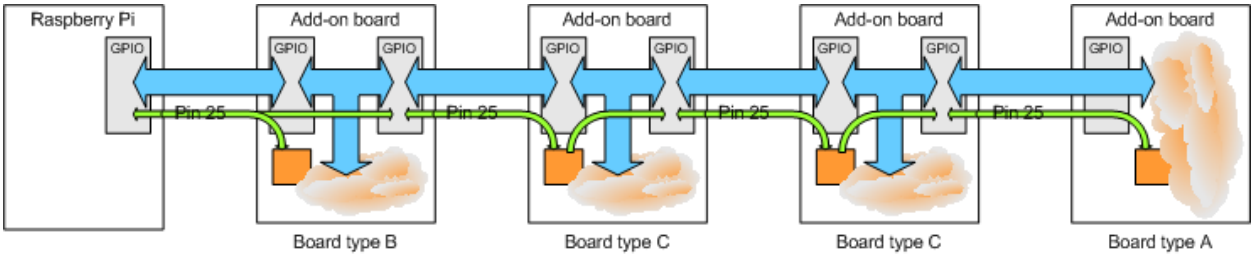
Various unique boards can be stacked together:



Multiple equal boards having GPIO pin 25 daisy-chained to male GPIO connector can be stacked:



Mixture of all board types can coexist:



3.2 Boards query and selection

One-pin communication protocol is introduced with the following key features:

- Bidirectional communication,
- Raspberry Pi is bus master, add-on boards are slaves,
- Communication is initiated by the Raspberry Pi.

3.2.1 Commands

		Required implementation		
Command	*Broadcast*	*Board type A*	*Board type B*	*Board type C*
Set backward compatibility	Yes	Yes	Yes	Yes
Reset	Yes	Yes	Yes	Yes
Initialize		Yes	Yes	Yes
Search	Yes	Yes	Yes	Yes
No operation	Yes	Yes	Yes	Yes
Select board		Optional	Optional	Yes
Read configuration		Yes	Yes	Yes
Write configuration		Yes	Yes	Yes
Lock mutex		Yes	Yes	Yes
Unlock mutex		Yes	Yes	Yes
Is mutex locked		Yes	Yes	Yes

3.2.1.1 Set backward compatibility

Syntax	0x00
Parameters	None
Response	None
Return values	None
Discussion	The command switches board to backward compatibility mode. Upon receiving this command, the board must set Board Selected flag (applicable to boards type C) and set their mutex flag to unlock state. Boards exit backward compatibility mode with *Reset* command. See also *Exception: Backward compatibility*.

3.2.1.2 Reset

Syntax	0x5A
Parameters	None
Response	None
Return values	None
Discussion	The command resets all boards' internal variables and states to power-on values. Daisy-chain position is also cleared. Next command to be executed is either *Search* or *Set backward compatibility*.

3.2.1.3 Initialize

Syntax	0x56 VendorID[3] ProductID[2] ChainLink
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor • ChainLink – one byte, specifies position of the board in the daisy chain to be selected; one-based index
Response	0xA5
Return values	Acknowledge
Discussion	The command initializes board's internal variables and states. It is expected that the board performs also initialization procedures for its components/modules. This command is executed also as master's acknowledge of a successfully received response to the *Search* command.
Example	Master initializes the 2nd board with ID 0xABCD made by vendor with ID 0x000123: 0x56 0x00 0x01 0x23 0xAB 0xCD 0x02

3.2.1.4 Search

Syntax	0x52
Parameters	None
Response	VendorID[3] ProductID[2]
Return values	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor
Discussion	This is command triggers boards to respond with their ID (VendorID and ProductID). The command is ignored by boards which already received Initialize command.

3.2.1.5 No operation

Syntax	0x50
Parameters	None
Response	None
Return values	None
Discussion	This command is never sent by the master. It mutates from a *Search* command. When a board with a daisy chain feature is not initialized yet and it receives a command bit stream starting with 0b010100 (the command is going to be either *Search* or *No operation*), it forces remaining two bits passed to next boards in the daisy chain to be 0. This mutation guarantees that only the first uninitialized board in the daisy chain responds to search query.

3.2.1.6 Select board

Syntax	0x90 VendorID[3] ProductID[2] ChainLink
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor • ChainLink – one byte, specifies position of the board in the daisy chain to be selected; one-based index
Response	0xA5
Return values	Acknowledge
Discussion	The command allows selection of a board in configuration where multiple equal boards are present. All further commands addressed with the VendorID and ProductID are processed only by the board selected. There is no deselect command. Currently selected board gets deselected when another board in the chain is selected. When boards type A and B contain elements that can have impact on another stacked boards performances, they should implement this command in order to disable/disconnect these elements from GPIO bus. See *Conflicts and malfunction prevention* for details.
Example	Master selects the 2nd board with ID 0xABCD made by vendor with ID 0x000123: 0x90 0x00 0x01 0x23 0xAB 0xCD 0x02

3.2.1.7 Read configuration

Syntax	0x92 VendorID[3] ProductID[2] DataLength
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor • DataLength – one byte, number of bytes in Data array; valid value between 1 and 32 inclusive
Response	Data[]
Return values	Data array, number of bytes equals to DataLength
Discussion	The command retrieves up to 32 bytes of stored configuration. The data content is not covered by this specification; it is up to the vendor to freely define the length and placeholders. The typical use case would be to use the board's sensor's calibration values.
Example	Master requests 5 bytes from the selected board with ID 0xABCD made by vendor with ID 0x000123: 0x92 0x00 0x01 0x23 0xAB 0xCD 0x05

3.2.1.8 Write configuration

Syntax	0x95 VendorID[3] ProductID[2] DataLength Data[]
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor • DataLength – one byte, number of bytes in Data array; valid value between 1 and 32 inclusive • Data – data array to be stored in the board
Response	0xA5
Return values	Acknowledge
Discussion	The command stores up to 32 bytes of configuration. The data content is not covered by this specification; it is up to the vendor to freely define the length and placeholders. The vendor can implement the configuration data, all or partially, to be read-only. Even if the complete configuration is read-only, the board returns acknowledge. When the configuration is stored on the board in slower non-volatile memory, the board should return acknowledge as a result of properly received command. It is recommended that the application checks the modification by performing *Read configuration* command few tenths of a second later. The typical use case would be defining the board's power-on state of outputs.
Example	Master writes 5 data bytes to the selected board with ID 0xABCD made by vendor with ID 0x000123: 0x95 0x00 0x01 0x23 0xAB 0xCD 0x05 0x44 0x44 0x44 0x44 0x44

3.2.1.9 Lock mutex

Syntax	0x99 VendorID[3] ProductID[2]
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor
Response	0xA5
Return values	Acknowledge
Discussion	The command sets a mutex flag value. The board operation is independent of this flag. Its purpose is to allow implementation of a mechanism to prevent applications race conditions.
Example	Master sets mutex flag for the selected board with ID 0xABCD made by vendor with ID 0x000123: 0x99 0x00 0x01 0x23 0xAB 0xCD

3.2.1.10 Unlock mutex

Syntax	0x9C VendorID[3] ProductID[2]
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor
Response	0xA5
Return values	Acknowledge
Discussion	The command clears a mutex flag value. The board operation is independent of this flag. Its purpose is to allow implementation of a mechanism to prevent applications race conditions. Power-on flag value is cleared. The flag is cleared also by the *Reset* command.
Example	Master clears mutex flag for the selected board with ID 0xABCD made by vendor with ID 0x000123: 0x9C 0x00 0x01 0x23 0xAB 0xCD

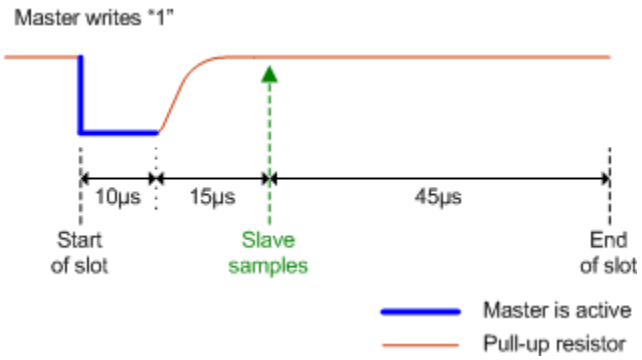
3.2.1.11 Is mutex locked

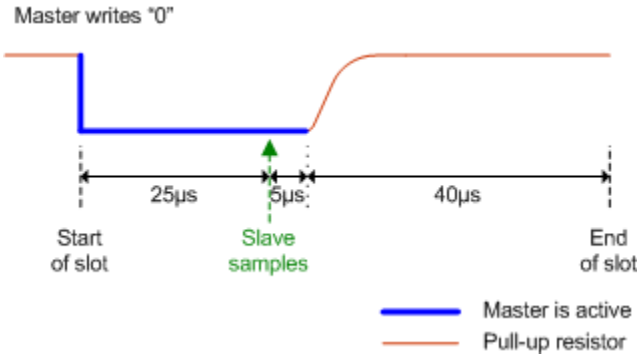
Syntax	0x9F VendorID[3] ProductID[2]
Parameters	• VendorID – 3-bytes identificator of the board's vendor • ProductID – 2-bytes code of the board, as defined by the vendor
Response	0xFF or 0xA5
Return values	When the mutex flag is set, returned value is 0xFF. Cleared flag is indicated by returning 0xA5.
Discussion	The command returns a mutex flag value. The board operation is independent of this flag. Its purpose is to allow implementation of a mechanism to prevent applications race conditions.
Example	Master clears mutex flag for the selected board with ID 0xABCD made by vendor with ID 0x000123: 0x9C 0x00 0x01 0x23 0xAB 0xCD

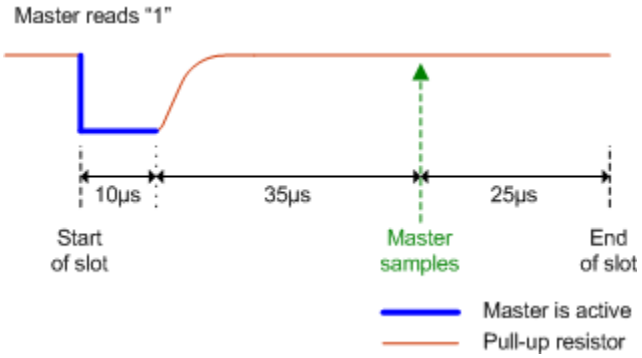
3.2.2 Timing

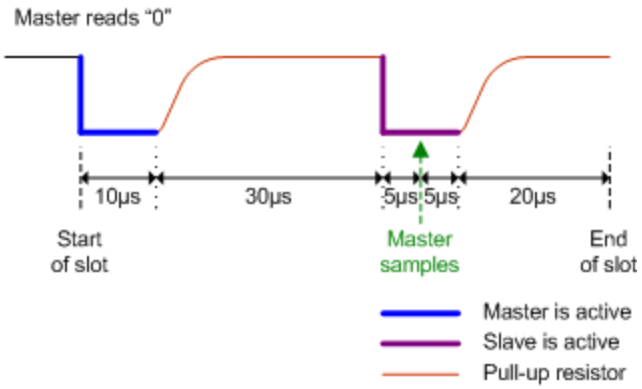
Base timing unit is 10µs. Tolerances are ±5%.

3.2.2.1 Bit time slot

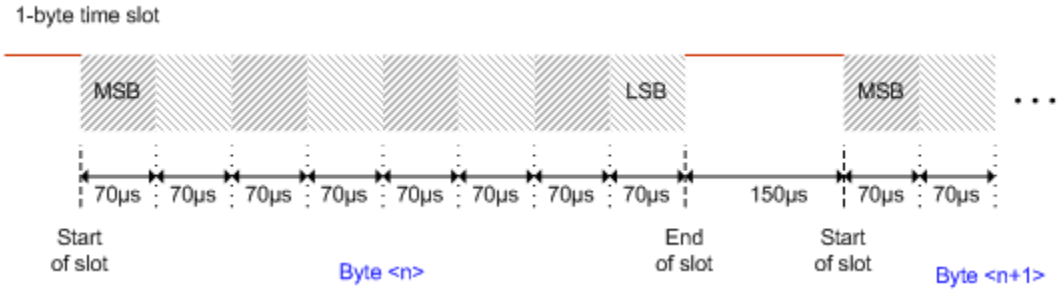




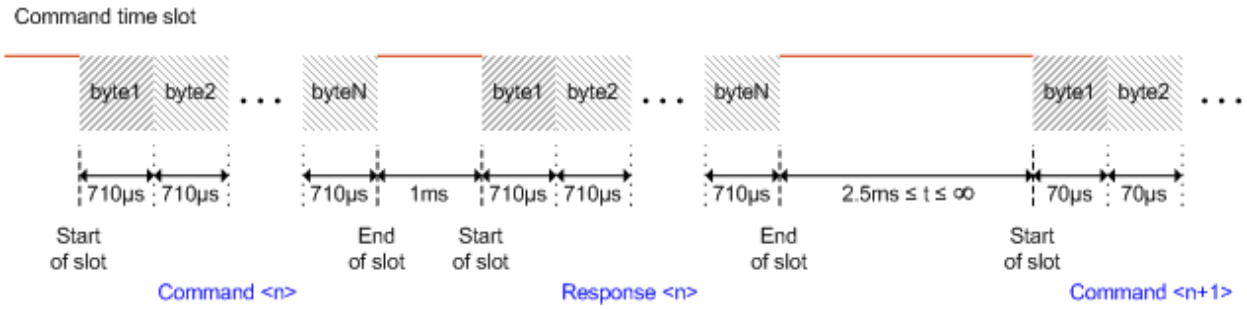




3.2.2.2 Byte time slot



3.2.2.3 Command time slot



3.2.2.4 Checksums

Each command and response packet contains a 2-bytes trailing CRC checksum.

CRC polynomial is $X^{16} + X^{12} + X^5 + 1$.

3.2.2.5 Exception: Backward compatibility

Backward compatibility can be activated with the *Set backward compatibility* command. Another option is to hold GPIO pin 25 permanently low.

3.3 Conflicts and malfunction prevention

By stacking various add-on boards, several potential issues are introduced.

After the Raspberry Pi detects add-on boards presence, it can perform various checks. The checks include:

- GPIO pins usage conflicts (eg. one board has pull-up resistor attached to one GPIO pin, where the other has pull-down resistor attached to this same pin.)
- Bus address conflicts (eg. more boards have the same I2C device with equal address defined)
- GPIO overcurrent case (eg. more boards have LED attached to the same GPIO pin)
- Overall power consumption of each board can be summarized and checked against maximum power consumption allowed.

A standalone tool should be implemented to perform these checks on user's request. Optionally, the tool could contain an option to check potential issues when a planned add-on board would be added to existing configuration.

The specification allows improved prevention actions performed by each board. If the board has implemented *Select board* command (mandatory for type C boards, optional for type A and B boards), then it can disable its various modules which could cause conflicts and/or malfunction of another stacked board.

Key role during the checking procedure has a global database. It contains related data details for each board following this specification. See *Global database* chapter for further details.

4 USB boards

To be defined…

5 Unified software support

There are two layers of software support:

- Pin/Port layer
- Data layer

The Pin/Port layer deals with signals and buses. It exposes interface to:

- Set features (eg. set pin direction, enable pull-ups)
- Set digital outputs
- Read digital inputs
- Set PWM outputs
- Write and read byte array to/from SPI bus
- Write and read byte array to/from I2C bus
- Write and read byte array to/from UART
- Sets analog outputs
- Reads analog inputs
- More to be defined…

The Data layer is a wrapper around the Pin/Port layer and deals with content. It exposes interface to:

- Sensors (get temperature, acceleration, humidity, light intensity, …)
- LCDs (display data)
- Various I/O chips
- More to be defined…

Further text to be added…

6 Global database

Global database has the following key purposes:

- Straight-forward unified support by software, each hardware feature is implemented only once
- Overview of all (currently) supported and unsupported features of the user's particular configuration
- Conflicts and malfunction detection / prevention when more boards are stacked
- Software support planning (overview about available add-on boards and missing (partial) support)

A record for each supported add-on board contains the following data:

- Vendor ID
- Product ID
- Type of the board (A/B/C)
- Is *Select board* command implemented (applicable to type A and B boards)
- Maximum overall current consumption
- I2C addresses occupied
- Are I2C devices disabled with *Select board* command (if applicable)
- For each GPIO data pin:
    - Pin type (input, output, bidirectional, OC, etc.)
    - Does *Select board* command switches it to hi-Z state
    - Maximum current sink at "1" (applicable when pin is input or bidirectional)
    - Maximum current sink at "0" (applicable when pin is input or bidirectional)
    - Value of pull-up resistor (when applicable)
    - Value of pull-down resistor (when applicable)
    - Can the pull resistor be disabled (applicable when *Select board* command is implemented)
    - Is an element with higher capacitance connected to the pin (eg. FET's Gate pin); limit capacitance to be defined…
- For each software-controllable module (relation to software's Data layer):
    - Module ID (according to Index of modules)
    - Parent Module ID (applicable eg. when the sensor chip is attached to the board internal I/O expander)
    - Software driver/handler configuration parameters (eg. pins or addresses)

Index of modules is a superset of modules/elements which any supported board contains. The following data is included for each module:

- Module ID
- Chip/Module (eg. sensor label)
- Software driver/handler ID (applicable when becomes supported)
- Version of the software when the module became supported

Records for USB devices to be defined…

7 Misc

GPIO pin 25 protocol can be handled by a dedicated chip or as a functionality of the board's main processor.

Vendor ID is assigned to the vendor by the global database maintainer.

When a vendor develops a new add-on board, it should provide data for the database record as a part of the development procedure.

For DIY hobbyists a dedicated Vendor ID could be assigned with Product ID matrix covering all kind of modules.