

# GPIO Protocol Specifications

*Description of the various commands and procedures*

## 1. Intro

This Gpio Protocol is used to configure, control and exchange data of GPIO elements connected to a microcontroller. It involves a host machine ( like a desktop computer) who is the **MASTER** which communicates to a microcontroller (**SLAVE**). For example, the master can tell the slave to set up an led or a button, and control it. This abstracts all the low level logic from the master, since the slave takes care of it.

### 1.1. Pin types

The slave has a set of GPIO pins which can have any number of the following type of pins:

1. **Digital input pins**
2. **Digital output pins**
3. **Analog input pins**

### 1.2. Nomenclature

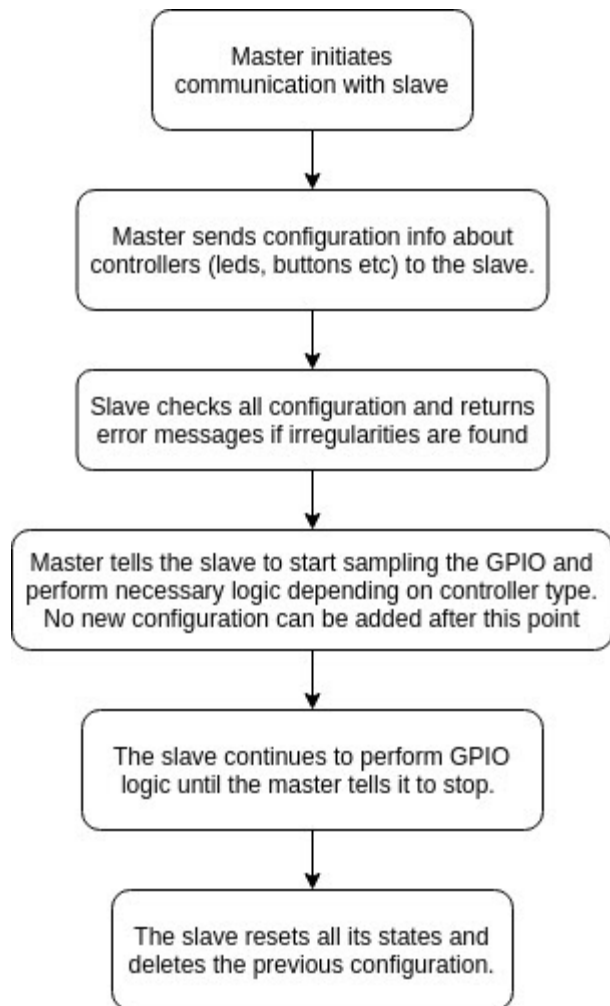
The following nomenclature and definitions are used in the document.

- **MASTER** : Used to denote the host machine (a high level computer) which initiates communication, sets up gpio elements etc.
- **SLAVE** : Used to denote the microcontroller where all the low level logic is implemented.
- **CTRL** : Abbreviation for **control** as in to control something
- **CTRLR** : Abbreviation for **controller**. A controller is a device which is connected to the GPIO pins. Eg an LED, Button, Switch etc
- **CTRLS** : Abbreviation for **controllers**. Used to denote a group of controllers.
- **HW\_TYPE**: Abbreviation for hardware type. Used to denote the type of controller.
- **GPIO\_NUM\_DIGITAL\_INPUT\_PINS** : Used to denote the number of digital input pins on the SLAVE.
- **GPIO\_NUM\_DIGITAL\_OUTPUT\_PINS** : Used to denote the number of digital output pins on the SLAVE.
- **GPIO\_NUM\_ANALOG\_INPUT\_PINS** : Used to denote the number of analog input pins on the SLAVE.
- **System tick rate** : The frequency at which I/O is performed.
- **Controller tick rate** : The frequency at which I/O of a controller is performed.
- **Timestamp** : The timestamp of the system tick rate in microseconds. This is basically a counter. More on that in the later sections.
- **Configuration State**: The slave is in configuration state, where new controllers such as leds, buttons etc can be added.

- **Runtime State** : The slave has received all necessary configuration and is now in a state where it is performing IO operations with the GPIO and the necessary logic of the Controllers.

### 1.3. Control Flow Summary

The general procedure can be summarized as follows:



### 1.4. Controllers and hardware types

A **Controller** is a device that is connected to the gpio. This can be leds, buttons, analog potentiometer etc. The slave contains the low level logic to handle these controllers. Types of Controllers are specified as a hardware type (**HW\_TYPE**). **It is important to note that the maximum allowed number of pins for a particular controller type is 25.** The following details the list of supported hw types for controllers.

HW_TYPE	Pin	Num	Data range	Configur
---------	-----	-----	------------	----------

	type	Pins		able data range?
GPIO_BINARY_OUTPUT	Digital Output	1 : 25	0 : (2 <sup>num pins used</sup> ) - 1	NO
GPIO_BINARY_INPUT	Digital Input	1 : 25	0 : 2 <sup>num pins used</sup> - 1	NO
GPIO_ANALOG_INPUT	Analog Input	1	0 : Resolution of ADC	YES
GPIO_STEPPED_OUTPUT	Digital Output	1 : 25	0 : num pins used	NO
GPIO_MUX_OUTPUT	Digital Output	1 : 10	1 : num pins used	NO
GPIO_N_WAY_SWITCH	Digital Input	1 : 25	1 : num pins used	NO
GPIO_ROTARY_ENCODER	Digital Input	2	Configurable	YES

#### 1.4.1. Binary Outputs (HW\_TYPE : GPIO\_BINARY\_OUTPUT)

A binary output type behaves such that its value is represented in binary form on its output pins. For eg: a single pin binary output can have values 0 or 1 in its pins. A two pin binary output can have output values 0x0 ,0x1, 0x2, 0x3 on its pins.

#### 1.4.2. Binary inputs (HW\_TYPE : GPIO\_BINARY\_INPUT)

A binary input is a very basic controller type. Its value represents the binary data present on its pins. For eg: a single pin button can have a value 0 or 1. A two pin dip switch can have the values 00, 01, 10, 11.

#### 1.4.3. Analog Input (HW\_TYPE : GPIO\_ANALOG\_INPUT)

This type is used to represent analog input controllers connected to any of the analog input pins. **It is important to note that each analog input controller should have only 1 pin.**

The client will automatically filter these inputs with a second order filter with a time constant of 20ms. This time constant can be set to any value above 0 during configuration.

#### 1.4.4. Stepped Output (HW\_TYPE : GPIO\_STEPPED\_OUTPUT)

This type is used to represent controllers where the intended behavior is akin to an LED ring. In other words, its value represents how many of its pins are switched to ON state. For example, if a stepped output has 8 pins and its value is 5, then its 1st, 2nd, 3rd, 4th and 5th pins are turned on.

#### 1.4.5. Mux output (HW\_TYPE : GPIO\_MUX\_OUTPUT)

This is a special type of controller, whose sole purpose is to enable/disable other controllers. Think of the logic as the output of a mux. Consider a case where there are 3 single pin binary outputs such as LEDs L1, L2 and L3 connected to the same pin on the slave, say pin number 1. Pins 2, 3, 4 of the slave are used to select which of the three leds glow. This is a common configuration when it comes to board layout as you end up using less pins.

A mux controller can be assigned to use pins 2,3,4 on the slave. It will then switch on only one pin for every tick and will iterate through its pins. So for every system tick, one of L1, L2 or L3 will be selected.

We can generalize it in the following way. If a mux has pin  $MuxPin_1, MuxPin_2 \dots MuxPin_N$  where  $N$  is the total number of pins assigned to it, the following state machine describes the output:

GPIO_MUX_OUTPUT state	$MuxPin_1$	$MuxPin_2$	$MuxPin_3$	.	.	.	$MuxPin_N$
0	ON	OFF	OFF	OFF	OFF	OFF	OFF
1	OFF	ON	OFF	OFF	OFF	OFF	OFF
2	OFF	OFF	ON	OFF	OFF	OFF	OFF
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
$N - 1$	OFF	OFF	OFF	OFF	OFF	OFF	ON

**The Mux Output Controller can have a maximum of 10 pins. Its state is incremented automatically and cannot be set from the MASTER.**

#### 1.3.6. N way Switch (HW\_TYPE : GPIO\_N\_WAY\_SWITCH)

Its value is reflective of which of its pins is in ON state. Only one of its pins is in ON state at any given time.

#### 1.4.7. Rotary Encoder (HW\_TYPE : GPIO\_ROTARY\_ENCODER)

This type represents a rotary encoder. Since it has infinite range, the range has to be set by the MASTER. **Rotary encoders need 2 pins, not more or less.** If more pins are assigned to it, an error is thrown.

### 1.5. Communication Methodology

This protocol follows a system where the **SLAVE** sends an acknowledgement for every packet sent by the **MASTER**, but not vice versa. i.e the **MASTER** does not send any acks for a packet sent from the **SLAVE**.

Whenever the master sends a query packet ( eg “get value of this controller” or “get board info”) the slave will first send an ack packet and then respond with a packet containing info for the query. **Note that the sequence number of the packet sent from the slave will always be 0.**

### 1.6. Packet Structure

The gpio packet size of 32 bytes in length and is comprised of the following fields as shown in the layout below.

Field	Size in bytes	Description
CMD	1	The main GPIO command field.
SUBCMD	1	For all the sub commands of the <b>CMD</b> .
CONTINUATION	1	Used in case the packet length is not enough to denote data. This field represents how many packets the data is split into.
RESERVED	1	Not used at the moment. Placed here in order to maintain word boundary alignment.
DATA	20	Contains data related to the <b>CMD</b> and <b>SUBCMD</b> .
TIMESTAMP	4	Used to denote the timestamp (in microseconds) of the packet.
SEQUENCE_NUMBER	4	denotes the sequence number of the packet, in order to ID it. This is used when replying to packets where the same sequence number is used when replying.

Field	Byte Offset
CMD	0

SUBCMD	1
CONTINUATION	2
RESERVED	3
DATA	4
	5
	6
	7
	8
	9
	0
	10
	11
	13
	14
	15
	16
	17
	18
	19
	20
	21
	22
	23
TIMESTAMP	24
	25
	26

	27
SEQUENCE_NUMBER	28
	29
	30
	31

## 1.7. Endianness

All packets are **LITTLE ENDIAN** in nature.

## 1.8. Timestamp

The timestamp is specified in micro seconds and starts counting from when the slave gets a GPIO\_START\_SYSTEM command and enters the Runtime state. It is like a counter and should be updated every tick period with the tick rate specified in microseconds.

## 1.8. Configuration and Runtime States for the slave

There are two states for the slave

1. **Configuration state** where new controllers, system settings etc can be set.
2. **Runtime state** of the slave, where GPIO logic and IO is performed. No new configuration can be added when the slave is in this state. The state is changed when the slave receives **GPIO\_START\_SYSTEM** command i.e after all configuration info has been sent to it.

The protocol does not allow for dynamic configuration of the system. **All configuration must be done before the slave is started with a GPIO\_START\_SYSTEM command!** Since runtime configuration is not supported by the protocol, the system needs to be reset and reconfigured again for new configurations.

## 2. List of CMD and descriptions

The following table denotes all the CMDs currently supported. Detailed description of each CMD's SUB\_CMD is provided in the next section.

CMD Value	String CMD	Description
1	GPIO_CMD_SYSTEM_CTRL	Command class to denote system control functionality. Eg : start, stop, reset

100	GPIO_CMD_CONFIG_CTRL	Command class to denote controller configuration functionality. Eg: add controller, set pins of a controller
101	GPIO_CMD_GET_VALUE	Command class used to send data from the <b>SLAVE</b> to the <b>MASTER</b>
102	GPIO_CMD_SET_VALUE	Command class used to send data from the <b>MASTER</b> to the <b>SLAVE</b>
250	GPIO_ACK	Used to denote Acknowledgement. The <b>SLAVE</b> responds with an ACK command packet for every packet from the <b>MASTER</b>

### 3. GPIO\_ACK command description

The ack command consists of the following packet structure. Note that the **SEQ\_NUM\_MASTER\_PACKET** corresponds to the seq num of the packet to which it is responding to. **Ack packets are only sent from the slave.**

#### 3.1. Slave → Master

Byte Offset		
0	CMD	ACK
1	SUBCMD	
4	SEQ_NUM_MASTER_PACKET	The sequence number of the packet it is acknowledging to.
5		
6		
7		
8	GPIO_RETURN_STATUS	A return code describing the outcome of the operation
24 -> 27	TIMESTAMP	The timestamp of the slave. <b>All ack packets sent before the slave is in Runtime state will have a timestamp of 0.</b>



## 3.2. GPIO\_RETURN\_STATUS description

Below is a list of all possible error codes.

GPIO_RETURN_STATUS Value	GPIO_RETURN_STATUS string value
0	GPIO_RETURN_OK
1	GPIO_RETURN_ERROR
2	GPIO_INVALID_CMD
3	GPIO_INVALID_SUB_CMD
4	GPIO_NO_CTRLRS_ADDED
5	GPIO_UNINITIALIZED_CTRLRS
6	GPIO_INVALID_TICK_RATE
7	GPIO_INVALID_CTRLR_ID
8	GPIO_INVALID_HW_TYPE
9	GPIO_INVALID_MUX_CTRLR
10	GPIO_INVALID_CTRLR_POLARITY
11	GPIO_NO_PINS_AVAILABLE
12	GPIO_INVALID_SHARING_OF_PINS
13	GPIO_RES_OUT_OF_RANGE
14	GPIO_UNRECOGNIZED_CMD
15	GPIO_PARAMETER_ERROR
16	GPIO_INVALID_CMD_FOR_CTRLR
17	GPIO_INVALID_RUNTIME_CONFIG

### 3.2.1. GPIO\_RETURN\_OK

Denotes that the command received was executed properly and everything is OK.

### 3.2.2. GPIO\_RETURN\_ERROR

Denotes that the command received could not be executed due to an unknown reason. This is just used to denote any undefined error codes that have not yet been implemented.

### 3.2.3. GPIO\_INVALID\_CMD

Denotes that the command received is not a valid CMD.

### 3.2.4. GPIO\_INVALID\_SUB\_CMD

Denotes that the sub command received is not a valid SUB\_CMD for that particular CMD

### 3.2.5. GPIO\_NO\_CTRLRS\_ADDED

Denotes that no controllers were added or configured. This is usually sent when a GPIO\_START\_SYSTEM sub command is received. More info in later sections.

### 3.2.6. GPIO\_UNINITIALIZED\_CTRLRS

Denotes that one or more controllers have not been completely initialized or configured properly. This is usually sent when a GPIO\_START\_SYSTEM sub command is received. More info in later sections.

### 3.2.7. GPIO\_INVALID\_TICK\_RATE

Denotes that the specified tick rate is not valid. This is used to denote an invalid tick rate for both, the system rate and controller tick rate.

### 3.2.8. GPIO\_INVALID\_CTRLR\_ID

Denotes that the controller id mentioned in a packet is invalid because of which a command could not be executed.

### 3.2.9. GPIO\_INVALID\_HW\_TYPE

Denotes that the packet contained an invalid or unsupported hardware type.

### 3.2.10. GPIO\_INVALID\_MUX\_CTRLR

Denotes that a mux controller info present in the received packet is invalid.

### 3.2.11. GPIO\_INVALID\_CTRLR\_POLARITY

Denotes that the controller's polarity specified in the received packet is invalid.

### 3.2.12. GPIO\_NO\_PINS\_AVAILABLE

Denotes that there are no available pins. This is a generic error type used to describe unavailability of any type of pins. For eg, Consider a system has digital output pins but no digital input pins. If a digital input controller is added and some pins are assigned to them, then this error is thrown.

### 3.2.13. GPIO\_INVALID\_SHARING\_OF\_PINS

Denotes that one or more pins of a controller are shared/duplicated with another controller. **Sharing of pins is allowed only if the controller is configured to be attached to a mux.**

### 3.2.14. GPIO\_RES\_OUT\_OF\_RANGE

Denotes that a resolution setting specified in the packet is not compatible with the controller. Usually used when the specified resolution is more than that of the ADC.

### 3.2.15. GPIO\_PARAMETER\_ERROR

Denotes that a value cannot be assigned to a controller. For eg, setting the value of a binary output controller type to more than its specified range.

### 3.2.16. GPIO\_INVALID\_CMD\_FOR\_CTRLR

Denotes that the specified command in the packet from the MASTER is not supported for a particular controller type.

### 3.2.16. GPIO\_INVALID\_RUNTIME\_CONFIG

Denotes that the specified command in the packet from the MASTER is not supported as the slave has already been configured and is in Runtime State. This happens when GPIO\_START\_SYSTEM command has been issued by the master prior and the slave cannot receive any new configuration commands. **Runtime configuration** is not supported by the protocol and for new configurations, the system needs to be reset and reconfigured again.

## 4. GPIO\_CMD\_SYSTEM\_CTRL description

The system control CMD has the following sub commands

CMD	SUB_CMD value	String SUB_CMD
	0	GPIO_STOP_RESET_SYSTEM

GPIO_CMD_SYSTEM_CTRL	1	GPIO_START_SYSTEM
	2	GPIO_SET_TICK_RATE
	3	GPIO_GET_BOARD_INFO

## 4.1. GPIO\_STOP\_RESET\_SYSTEM sub command

- stops the **SLAVE** from any I/O with the gpio controllers
- removes all previously configured controllers
- essentially puts the system to a **reset** state
- Can be sent when the slave is in pre or post configuration states

### 4.1.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_SYSTEM_CTRL
1	SUBCMD	GPIO_STOP_RESET_SYSTEM
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 4.1.2. Slave → Master GPIO\_RETURN\_STATUS

- This command can be sent when the slave is either Runtime or Configuration State.
- The slave sends an ack packet with a corresponding error code. **Note that the return code will always be GPIO\_RETURN\_OK** because nothing can go wrong, unless the slave is not responsive.

### 4.1.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp
- 0 if it is in Configuration State

## 4.2. GPIO\_START\_SYSTEM

- Informs the slave to start the GPIO I/O process.
- Done after configuration of controllers are complete.
- Can be sent only when the slave is in configuration state
- Upon success, the slave enters Runtime state

#### 4.2.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_SYSTEM_CTRL
1	SUBCMD	GPIO_START_SYSTEM
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

#### 4.2.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS and with a TIMESTAMP = 0.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	Start was successful
GPIO_NO_CTRLIS_ADDED	Cannot start the system since no controllers were added during configuration.
GPIO_UNINITIALIZED_CTRLIS	Cannot start the system since one or more controllers were not initialized/configured fully. Eg: a controller without any pins assigned to it.
GPIO_INVALID_SHARING_OF_PINS	Cannot start the system since one or more pins are shared between two or more controllers. Sharing is allowed only if they are attached to a mux output controller
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

#### 4.2.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ack packet with GPIO\_INVALID\_RUNTIME\_CONFIG return status.
- 0 if it is in Configuration State

### 4.3. GPIO\_SET\_TICK\_RATE

Sets the system tick rate of the SLAVE. The tick rate is the frequency at which I/O is performed with the controllers. This can be only sent when the slave is in runtime state. Tick rate can be set to the following possible values:

GPIO_SYSTEM_TICK_RATE_HZ	String Description
100	100 Hz
500	500 Hz
1000 (default)	1000 Hz
5000	5000 Hz

#### 4.3.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_SYSTEM_CTRL
1	SUBCMD	GPIO_SET_TICK_RATE
4	DATA	GPIO_SYSTEM_TICK_RATE_HZ
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

#### 4.3.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS. The TimeStamp is a unique number if this command is issued when the slave is in Runtime state. Else TimeStamp = 0.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The system is configured to run with new tick rate
GPIO_INVALID_TICK_RATE	Specified tick rate was not valid.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 4.3.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp
- 0 if it is in Configuration State

## 4.4. GPIO\_GET\_BOARD\_INFO

This command is used by the master to enquire the board information of the slave, that is, the number of gpio pins it contains. This is usually done in Configuration State as to know the information of the board before configuring it, but can be sent when the slave is in Runtime state as well. This is to accommodate the fact that different boards have different numbers of pins. The slave returns the following info

1. Number of digital input pins.
2. Number of digital output pins
3. Number of analog input pins.
4. Resolution of ADC in number of bits. Eg: 10 or 8. **If no adc (and consequently no analog input pins) exist on the board, this number will be equal to 0.**

#### Note:

The slave responds in the following sequence

1. First it sends an ACK packet with a timestamp (as described below).
2. Then it sends a packet containing the board information containing the same timestamp.

### 4.4.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_SYSTEM_CTRL
1	SUBCMD	GPIO_GET_BOARD_INFO
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 4.4.2. Slave → Master GPIO\_RETURN\_STATUS

1. The slave first sends an ack packet with a return status of **GPIO\_RETURN\_OK** and a timestamp.
2. Then it sends a packet containing the board info as shown in the packet structure below:

Byte Offset		
0	CMD	GPIO_CMD_SYSTEM_CTRL
1	SUBCMD	GPIO_GET_BOARD_INFO

4	DATA	Number of Digital input pins on the board
5		Number of Digital output pins on the board
6		Number of Analog input pins on the board
7		Resolution of ADC in bits (0 if no adc or analog pins exist)
24 -> 27	TIMESTAMP	The timestamp of the slave which is the same as the ACK Packet
28 → 31	SEQUENCE_NUMBER	0

#### 4.4.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp
- 0 if it is in Configuration State

## 5. GPIO\_CMD\_CONFIG\_CTRLs description

This gpio command is used to set up and configure controllers on the board. It has the following sub commands. Certain commands can be sent when the slave is in Runtime State as shown below.

CMD	SUB_CMD value	String SUB_CMD
GPIO_CMD_CONFIG_CTRLs	0	GPIO_RESET_ALL_CTRLs
	1	GPIO_RESET_CTRLR
	2	GPIO_ADD_CTRLR
	3	GPIO_ATTACH_CTRLR_TO_MUX
	4	GPIO_SET_CTRLR_POLARITY
	5	GPIO_SET_INPUT_CTRLR_TICK_RATE
	6	GPIO_SET_INPUT_CTRLR_NOTIF_MODE
	7	GPIO_ADD_PINS_TO_CTRLR
	8	GPIO_MUTE_UNMUTE_CTRLR



	10	GPIO_SET_ANALOG_CTRLR_RES
	11	GPIO_SET_CTRLR_RANGE
	12	GPIO_SET_CTRLR_DEBOUNCE_MODE
	13	GPIO_SET_ANALOG_CTRLR_TIME_CONSTANT

## 5.1. GPIO\_RESET\_ALL\_CTRLR

This sub command is used to reset data **of all configured controllers**. Note that it does not mean that it removes all configured controllers, but only resets their input/output values to the default values. This command can be sent during Runtime state **and** Configuration State.

### 5.1.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLR
1	SUBCMD	GPIO_RESET_ALL_CTRLR
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.1.2. Slave → Master GPIO\_RETURN\_STATUS

The slave will always return **GPIO\_RETURN\_OK**.

### 5.1.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp
- 0 if it is in Configuration State

## 5.2. GPIO\_RESET\_CTRLR

This sub command is used to reset the data of a particular controller. **Note that the controller should have been already added and configured.** This command can be sent during Runtime state **and** Configuration State.

### 5.2.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLR

1	SUBCMD	GPIO_RESET_CTRLR
4	DATA	CTRLR_ID of the controller which has to be reset
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.2.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller was reset successfully
GPIO_INVALID_CTRLR_ID	Specified controller ID was not found in the list of configured controllers.

### 5.2.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp
- 0 if it is in Configuration State

## 5.3. GPIO\_ADD\_CTRLR

This sub command is used to add a new controller, of a specific hardware type. The list of hardware types is specified in section 1.3. Note that a controller of a specific hardware type can only be added if its relevant pins exist on the board. For eg, an analog controller can be only added if there are analog pins on the board.

This command cannot be sent once the slave is in RunTime state i.e a new controller cannot be added once the configuration is finished and a GPIO\_START\_SYSTEM command is issued by the master.

### 5.3.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLR
1	SUBCMD	GPIO_ADD_CTRLR
4	DATA	CTRLR_ID of new controller
5		HW_TYPE of the new controller
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.3.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The new controller was added successfully
GPIO_NO_PINS_AVAILABLE	The new controller could not be added since the pins it uses either do not exist or have been used up.
GPIO_INVALID_CTRLR_ID	The new controller ID specified in the master packet already exists (has already been added previously)
GPIO_INVALID_HW_TYPE	The specified hw type in the master's packet is invalid.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.3.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.4. GPIO\_ATTACH\_CTRLR\_TO\_MUX

Attaches a controller to a mux controller of type GPIO\_MUX\_OUTPUT. **Do note that this command can only be sent when the slave is in Configuration State.** The following conditions have to be met

1. The mux output controller should have already been added and its pins defined.
2. The hw type of the controller which is to be attached to mux should be of compatible type. Not all hw types can be attached. This usually includes controllers such as rotary encoders, where its input signal is in the form of transitions. See the table below for supported hw types

HW_TYPE	Can it be attached to a mux controller?
GPIO_BINARY_OUTPUT	YES
GPIO_BINARY_INPUT	YES
GPIO_ANALOG_INPUT	YES

GPIO_STEPPED_OUTPUT	YES
GPIO_MUX_OUTPUT	NO
GPIO_N_WAY_SWITCH	YES
GPIO_ROTARY_ENCODER	NO

#### 5.4.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_ATTACH_CTRLR_TO_MUX
4	DATA	CTRLR_ID : id of controller which has to be attached to the mux controller
5		MUX_CTRLR_ID : id of the mux controller
6		ASSOCIATED_MUX_PIN : the pin number which “selects” the controller that has to be attached..
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

#### 5.4.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller was successfully attached to the specified mux controller.
GPIO_INVALID_MUX_CTRLR	The specified mux was not added, or the specified associated mux pin is not part of the mux controller, or the mux pin is already associated with another controller id
GPIO_INVALID_CMD_FOR_CTRLR	The controller which is to be attached to the mux controller had an incompatible hw type
GPIO_INVALID_CTRLR_ID	The controller which is to be attached to the mux was not found. This can be attributed to the fact that the controller may not have been added or

	the controller id specified was incorrect.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.4.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.5. GPIO\_SET\_CTRLR\_POLARITY

Sets the polarity of a controller. **Do note that this command can only be sent when the slave is in Configuration State.** The polarity can be either active high or active low. This is applicable to only select hw types.

CTRLR_POLARITY	Description
GPIO_ACTIVE_HIGH (default)	Denotes that the controller's I/O is active high in nature. <b>Note that all controllers by default are configured as active high</b>
GPIO_ACTIVE_LOW	Denotes that the controller's i/O is active low in nature

HW_TYPE	Configurable polarity?
GPIO_BINARY_OUTPUT	YES
GPIO_BINARY_INPUT	YES
GPIO_ANALOG_INPUT	NO
GPIO_STEPPED_OUTPUT	YES
GPIO_MUX_OUTPUT	YES
GPIO_N_WAY_SWITCH	YES
GPIO_ROTARY_ENCODER	NO

### 5.5.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_SET_CTRLR_POLARITY
4	DATA	CTRLR_ID : id of controller whose polarity has to be changed
5		CTRLR_POLARITY : the new polarity
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.5.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller's polarity was successfully changed
GPIO_INVALID_CMD_FOR_CTRLR	The controller whose polarity was to be changed is of incompatible hw type.
GPIO_INVALID_CTRLR_ID	The controller whose polarity is to be changed is not found. This can be attributed to the fact that the controller may not have been added or the controller id specified was incorrect.
GPIO_INVALID_CTRLR_POLARITY	Invalid controller polarity
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.5.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in a Configuration State.

## 5.6. GPIO\_SET\_INPUT\_CTRLR\_TICK\_RATE

Sets the tick rate of a controller whose hw type is input in nature. **Do note that this command can only be sent when the slave is in Configuration State.**

The controller tick rate is specified in multiples of the system tick rate. The default tick rate of a controller is 1.

HW_TYPE	Configurable tick rate?
GPIO_BINARY_OUTPUT	NO
GPIO_BINARY_INPUT	YES
GPIO_ANALOG_INPUT	YES
GPIO_STEPPED_OUTPUT	NO
GPIO_MUX_OUTPUT	NO
GPIO_N_WAY_SWITCH	YES
GPIO_ROTARY_ENCODER	NO

#### 5.6.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_SET_INPUT_CTRLR_TICK_RATE
4	DATA	CTRLR_ID : id of controller whose tick rate has to be changed
5		DELTA_TICK_RATE : the new tick rate of the controller specified as multiples of the system tick rate.
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

#### 5.6.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller's tick rate was successfully changed
GPIO_INVALID_CMD_FOR_CTRLR	The controller whose tick rate was to be changed is of incompatible hw type.
GPIO_INVALID_CTRLR_ID	The controller whose tick rate is to be changed is not found amongst compatible hw types. This can

	be attributed to the fact that the controller may not have been added or the controller id specified was incorrect.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.6.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.7. GPIO\_SET\_INPUT\_CTRLR\_NOTIF\_MODE

Sets the notification mode of an input controller. **Do note that this command can only be sent when the slave is in Configuration State.** The notification mode describes the circumstances in which an input controller notifies the master about its value, using a GPIO\_CMD\_GET\_VALUE packet. The following table describes the various notification modes.

GPIO_NOTIF_MODE	Description
ON_VALUE_CHANGE (default)	A GPIO_CMD_GET_VALUE packet is sent to the master each time the controller's value changes.
EVERY_CTRLR_TICK	A GPIO_CMD_GET_VALUE packet is sent to the master every controller tick. This is akin to continuous mode.
WHEN_TOGGLED_ON	<p>A GPIO_CMD_GET_VALUE packet is sent to the master each time the controller's value is switched from OFF → ON → OFF. What constitutes as ON or OFF depends on the controller's polarity.</p> <p>Not applicable to multi pin GPIO_BINARY_INPUT, multi pin GPIO_STEPPED_OUTPUT and rotary encoders</p>
WHEN_TOGGLED_OFF	A GPIO_CMD_GET_VALUE packet is sent to the master each time the controller's value is switched from ON → OFF → ON. What constitutes as ON or OFF depends on the controller's polarity.



	Not applicable to multi pin GPIO_BINARY_INPUT, multi pin GPIO_STEPPED_OUTPUT and rotary encoders
--	--

HW_TYPE	Configurable notification mode?
GPIO_BINARY_OUTPUT	N/A
GPIO_BINARY_INPUT	YES
GPIO_ANALOG_INPUT	YES
GPIO_STEPPED_OUTPUT	N/A
GPIO_MUX_OUTPUT	N/A
GPIO_N_WAY_SWITCH	YES
GPIO_ROTARY_ENCODER	YES

#### 5.7.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_SET_INPUT_CTRLR_NOTIF_MODE
4	DATA	CTRLR_ID : id of controller whose notification mode has to be changed
5		GPIO_NOTIF_MODE : the new notif mode of the controller.
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

#### 5.7.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller's notification mode was successfully changed
GPIO_INVALID_CMD_FOR_CTRLR	The controller whose notification mode was to be changed is of incompatible hw type.
GPIO_INVALID_CTRLR_ID	The controller whose notification mode is to be

	changed is not found amongst compatible hw types. This can be attributed to the fact that the controller may not have been added or the controller id specified was incorrect.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.7.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.8. GPIO\_ADD\_PINS\_TO\_CTRLR

Adds pins to a controller. **Do note that this command can only be sent when the slave is in Configuration State.**

### 5.8.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLR
1	SUBCMD	GPIO_ADD_PINS_TO_CTRLR
4	DATA	CTRLR_ID : id of controller to which pins have to be added
5		NUM_PINS_IN_PACKET : The number of pins <b>whose info is specified in this packet.</b>
6 → 23		PINS : The pin numbers
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.8.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The pins have been successfully added to the controller
GPIO_INVALID_CTRLR_ID	The specified controller id does not exist or has not been added

GPIO_INVALID_CMD_FOR_CTRLR	The specified controller already has pins assigned to it or too many pins have been specified for this controller.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.8.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.9. GPIO\_MUTE\_UNMUTE\_CTRLR

Command to mute or unmute a controller.

If an output hw type controller is muted, then it will be in OFF state no matter what value is sent by the host to the controller. The current values are stored internally when muted and will be restored when unmuted. This also applies to new values sent when the controller is muted.

If an input hw type is muted, then it will not notify the host about its value (irrespective of its notification mode). Only certain hw types support. Can be used when the slave is in Runtime or Configuration States.

GPIO_CTRLR_MUTE_STATUS	Description
GPIO_CTRLR_MUTED	Controller is muted
GPIO_CTRLR_UNMUTED (default)	Controller is unmuted. This should be the default state of all controllers.

HW_TYPE	Can be muted/unmuted?
GPIO_BINARY_OUTPUT	YES
GPIO_BINARY_INPUT	YES
GPIO_ANALOG_INPUT	YES
GPIO_STEPPED_OUTPUT	YES
GPIO_MUX_OUTPUT	NO

GPIO_N_WAY_SWITCH	YES
GPIO_ROTARY_ENCODER	YES

### 5.9.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_MUTE_UNMUTE_CTRLR
4	DATA	CTRLR_ID : id of controller whose mute status has to be changed
5		GPIO_CTRLR_MUTE_STATUS: the new mute status of the controller
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.9.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller's mute state was successfully changed
GPIO_INVALID_CTRLR_ID	The controller whose mute status is to be changed is not found amongst compatible hw types. This can be attributed to the fact that the controller may not have been added or the controller id specified was incorrect.

### 5.9.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp.
- 0 if it is in Configuration State

## 5.10. GPIO\_SET\_ANALOG\_CTRLR\_RES

Set the resolution in bits of an analog input controller. The new resolution should follow the rule:

$$1 \leq AnalogController_{resolution} \leq ADC_{resolution}$$

By default, the analog input controller's resolution should be equal to that of the ADC. **Do note that this command can only be sent when the slave is in Configuration State.**

### 5.10.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_SET_ANALOG_CTRLR_RES
4	DATA	CTRLR_ID : id of the analog input controller whose resolution has to be changed
5		ANALOG_CTRLR_RES: the new resolution of the controller.
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.10.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The analog input controller's resolution has successfully changed.
GPIO_INVALID_CTRLR_ID	The analog input with the specified controller id is not found amongst the configured controller list or when there are no analog pins available on the board
GPIO_RES_OUT_OF_RANGE	The resolution specified in the master packet is more than the supported resolution of the board
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.10.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.11. GPIO\_SET\_CTRLR\_RANGE

Set the range for a controller's value. **The slave will clip values higher or lower the specified range.** Only certain HW types support this feature. **It is necessary to set the controller range for rotary encoders.**

**Do note that this command can only be sent when the slave is in Configuration State.**

HW_TYPE	Configurable range?
GPIO_BINARY_OUTPUT	NO
GPIO_BINARY_INPUT	NO
GPIO_ANALOG_INPUT	YES (default min = 0, default max = $2^{\text{adc\_res}} - 1$ )
GPIO_STEPPED_OUTPUT	NO
GPIO_MUX_OUTPUT	NO
GPIO_N_WAY_SWITCH	NO
GPIO_ROTARY_ENCODER	YES (default min = 0, default max = 0)

#### 5.11.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLIS
1	SUBCMD	GPIO_SET_ANALOG_CTRLR_RES
4	DATA	CTRLR_ID : id of the controller whose resolution has to be changed
5		Empty for word alignment
6		Empty for word alignment
7		Empty for word alignment
8 → 11		MIN_VAL : 32 bit integer representing the minimum possible value.
12 → 15		MAX_VAL : 32 bit integer representing the maximum possible value.
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

#### 5.11.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The range of the controller was successfully set.

GPIO_INVALID_CTRLR_ID	The specified controller id is not found amongst the configured controller list.
GPIO_INVALID_CMD_FOR_CTRLR	Indicates that the controller specified in the master packet does not support different ranges.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

### 5.11.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.12 GPIO\_SET\_CTRLR\_DEBOUNCE\_MODE

Enable or disable sw debouncing for the input pins of a digital input hw type controller. By default it is always enabled. The debounce mode is denoted by a single byte as follows. **Do note that this command can only be sent when the slave is in Configuration State.**

GPIO_DEBOUNCE_MODE	Value
GPIO_DEBOUNCE_ENABLED	0
GPIO_DEBOUNCE_DISABLED	1

HW_TYPE	Configurable Debounce?
GPIO_BINARY_OUTPUT	NO
GPIO_BINARY_INPUT	YES
GPIO_ANALOG_INPUT	NO
GPIO_STEPPED_OUTPUT	NO
GPIO_MUX_OUTPUT	NO

GPIO_N_WAY_SWITCH	YES
GPIO_ROTARY_ENCODER	NO

### 5.12.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_SET_CTRLR_DEBOUNCE_MODE
4		CTRLR_ID : id of the controller whose debounce mode to be changed
5		GPIO_DEBOUNCE_MODE
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.12.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The debounce of the controller was successfully enabled/dsiabled.
GPIO_INVALID_CTRLR_ID	The specified controller id is not found amongst the configured & compatible controller list.
GPIO_INVALID_CMD_FOR_CTRLR	Indicates that the controller specified in the master packet does not support debounce modes.
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.



### 5.11.3. Slave → Master Timestamp

- If it is in Runtime State, the slave will send a unique timestamp in the ACK packet containing the GPIO\_INVALID\_RUNTIME\_CONFIG gpio return status.
- 0 if it is in Configuration State

## 5.13 GPIO\_SET\_ANALOG\_CTRLR\_TIME\_CONSTANT

This command is to set the time constant of the filter of an analog controller. **By default the time constant is 0.020 seconds.** The new time constant has to be above 0. This command can only be used during configuration state.

### 5.13.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_SET_ANALOG_CTRLR_TIME_CONSTANT
4		CTRLR_ID : id of the analog controller whose time constant has to be changed
5 - 7		Reserved
8-12		Time constant in seconds( Float)
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 5.13.2. Slave → Master GPIO\_RETURN\_STATUS

The slave sends an ack packet with any of the following possible GPIO\_RETURN\_STATUS.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The new time constant was successfully set.
GPIO_INVALID_CTRLR_ID	The specified controller id is not found amongst the analog controller list.
GPIO_INVALID_CMD_FOR_CTRLR	Indicates that the controller specified was not in the analog controller list but in another controller list.

GPIO_PARAMETER_ERROR	The time constant specified is less than or equal to 0
GPIO_INVALID_RUNTIME_CONFIG	The slave has already been started since it has received a prior GPIO_START_SYSTEM command and is in Runtime State.

## 6. GPIO\_CMD\_GET\_VALUE description

This command is used to get controller values from the slave and can be used in two scenarios:

1. When the slave has to notify the master of a controller's value, it sends a GPIO\_CMD\_GET\_VALUE packet.
2. The master can request the slave for a particular controller's value. **The slave responds by first sending an ACK packet with a return status.** If the controller id specified is valid, then another packet with the value of the controller is sent from the slave.

The Master can ask the value of any type of hw type except a GPIO\_MUX\_OUTPUT.

### 6.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_GET_VALUE
1	SUBCMD	0x00
4	DATA	CTRLR_ID : The id of the controller whose value is requested.
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

### 6.2. Slave → Master Packet Structure

If the master has requested for the value, then the slave first sends an ack packet with the following possible gpio return status and a timestamp.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller id specified in the master packet is valid and its value will be sent in the <b>next</b> packet.
GPIO_INVALID_CTRLR_ID	The specified controller id is not found amongst

	the configured and compatible controller list.
--	--

After sending an ack packet with GPIO\_RETURN\_OK, the slave sends a packet containing the controller value as shown in the packet structure below, and with the same timestamp as that of the ack packet. **Note that this packet structure is the same when the slave sends the value of a controller without the master requesting it.**

Byte Offset		
0	CMD	GPIO_CMD_GET_VALUE
1	SUBCMD	0x00
4	DATA	CTRLR_ID : The id of the controller whose value is requested or has to be sent in order to notify the master.
5 → 7		RESERVED : Empty bytes in order to preserve word boundary
8 → 11		CTRLR_VALUE : The value of the controller
24 → 27	TIMESTAMP	The timestamp of the slave.
28 → 31	SEQUENCE_NUMBER	0x00

## 7. GPIO\_CMD\_SET\_VALUE description

This command is used by the master to set a value of a particular controller.

- If this command is sent during Configuration state, then this value is the default start value of the controller. This will be supported by all hw types with the exception of GPIO\_MUX\_OUTPUT.
- If this command is sent Runtime state, only hardware types of digital output will support this command.

### 7.1. Master → Slave Packet Structure

Byte Offset		
0	CMD	GPIO_CMD_SET_VALUE
1	SUBCMD	0x00
4	DATA	CTRLR_ID : The id of the controller whose value is has to be set.

5 → 7		RESERVED : Empty bytes in order to preserve word boundary
8 → 11		CTRLR_VALUE : The value of the controller
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet.

## 7.2. Slave → Master Packet Structure

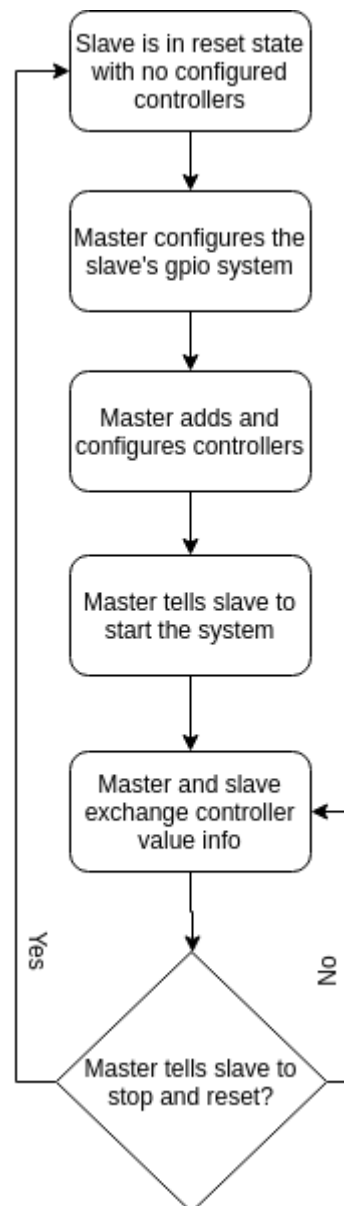
The slave sends an ack packet with the following possible gpio return status and a unique timestamp.

Possible GPIO_RETURN_STATUS	Description
GPIO_RETURN_OK	The controller id specified in the master packet is valid and its value is set to the new one.
GPIO_INVALID_CTRLR_ID	The specified controller id is not found amongst the configured controller list.
GPIO_PARAMETER_ERROR	The value sent is not in the range of the controller or is not compatible with it.

## 8. Command Sequence with Example

The setup sequence is divided into three main stages:

1. System configuration
2. Controller configuration
3. Exchanging controller value information between master and slave. This is similar to a runtime stage.



For example, consider a setup where the following controllers that have to be configured:

1. A simple LED (binary output) having an ID of 1 connected on a digital output pin 1.
2. Two LED rings of IDs 2 and 3. Each has 5 LEDs (stepped output) connected to the same digital output pins 2,3,4,5 and 6. The individual LED ring is selected using digital output pins 7 and 8.
3. A simple button having ID 4 on digital input pin 1
4. A four way switch having ID 5 on digital input pins 2,3,4 and 5

## 8.1. System setup

### 8.1.1. Stop and reset the system

First the Master should stop and reset any previous configuration to ensure that the slave is in reset state. This is done by sending a **GPIO\_STOP\_RESET\_SYSTEM** packet. The slave sends an ack packet with the return status **GPIO\_RETURN\_OK**.

### 8.1.2. Get board information

The master queries the slave for the board information by sending a **GPIO\_GET\_BOARD\_INFO** packet. The slave responds by first sending an ack packet with return status **GPIO\_RETURN\_OK** followed by the packet containing board info. The master can use this information to validate the configuration requirements and see if it can be supported on the slave's gpio capabilities.

### 8.1.3. Set System Tick Rate:

By default, the tick rate of the slave's system at reset will be 1000Hz. You can change the system tick if needed by using the **GPIO\_SET\_TICK\_RATE** command.

## 8.2. Controller configuration

### 8.2.1. Adding the single LED (ID = 1)

The LED is of hw type **GPIO\_BINARY\_OUTPUT**. First the controller is added using the **GPIO\_ADD\_CTRLR** command. The slave responds with return status **GPIO\_RETURN\_OK**.

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS
1	SUBCMD	GPIO_ADD_CTRLR
4	DATA	CTRLR_ID : 0x01
5		HW_TYPE : 0x01
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

Then the master adds pins to this controller using **GPIO\_ADD\_PINS\_TO\_CTRLR** command. The slave responds with return status **GPIO\_RETURN\_OK**.

Byte Offset		
0	CMD	GPIO_CMD_CONFIG_CTRLRS

1	SUBCMD	GPIO_ADD_PINS_TO_CTRLR
4	DATA	CTRLR_ID : 0x01
5		NUM_PINS_IN_PACKET : 0x01
6		PINS : 0x01
28 → 31	SEQUENCE_NUMBER	The sequence number of the packet

After this the master can then configure other parameters of the controller such as its mute status etc.

**TO BE CONTINUED.....**