gpio base Raspi CM4 Data Sheet

Raspberry Pi Compute Module 4 Expansion Board with IEEE1588 Gigabit-Ethernet, IEEE 802. 11b/g/n/ac WIFI, HDMI, 3 USB Ports and 6 configurable Interfaces

Features

Interfaces and external signals

- IEEE1588 Gigabit-Ethernet
- 2,4 GHz and 5,0 GHz IEEE 802. 11b/g/n/ac WIFI
- 2.4 Ghz radio
- 3 USB 2.0 Ports
- HDMI 2.0 Port
- Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM
- Options for 8GB, 16GB, or 32GB eMMC Flash memory
- Up to six IO ports/serial communication interfaces, each multiplexed over a Microchip SAME54 co-processor and configurable to operate as either:
  - UART
  - I2C
  - SPI
  - digital IO
  - taskit gpio module interface
- Micro USB socket for updating the CM4
- Raspberry Pi HAT connector

SAME54 Co Processor

- Microchip SAME54N20A Processor
- ARM Cortex-M4F Core
- 120 MHz
- 1 MB Flash
- 256 KB SRAM
- CoAP (Constrained Protocol) REST-ful API with HTTP interface over tsbcoap software
- AT Command API
- Demultiplexing over MQTT protocol with taskit tsbmqtt software

Power Supply

- +9 to +24V DC
- PoE (Power over Ethernet)

Mechanical Characteristics

- Dimensions: 100 x 160 x 25 mm

Portux CM4: Technical Reference v1.1

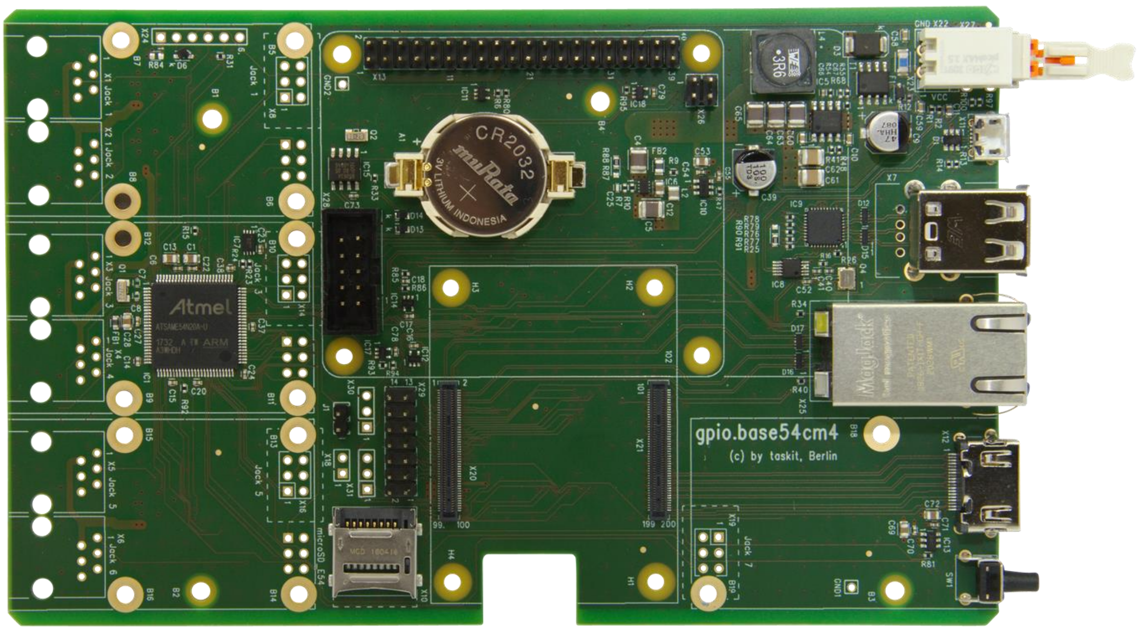
gpio.base54cm4
(c) by taskit, Berlin
Atmel
CR2032
muRata
Jack 1
Jack 2
Jack 3
Jack 4
Jack 5
Jack 6
Jack 7
microSD

Table of Contents

List of Figures

List of Tables

1. Hardware Description

1.1. USB

The Universal Serial Bus interface (USB) complies with the Universal Serial Bus (USB) 2.0 specification and supports up to 480MBps signaling.

1.2. Ethernet

The CM4 has an onboard Gigabit Ethernet PHY — the Broadcom BCM54210PE — some of the major features of this PHY include:

- IEEE 1588-2008 compliant
- Detection and correction of swapped pairs
- MDI crossover, pair skew and pair polarity correction

1.3. WIFI

Optional Dual-band 2.4/5.0GHz IEEE 802.11 b/g/n/ac wireless LAN have modular compliance certification. This allows the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

The two CM4 module connectors are positioned so the onboard wireless antenna is at the edge of the board for best wireless performance. Alternatively there is a standard U.FL connector on the module, so that an external antenna can be used.

1.4. 2.4 Ghz radio

Optional 2.4 Ghz radio.
As with the WIFI option, you can choose between external and internal antenna.
The module can be switched on or off to reduce the current.

1.5. HDMI

The CM4 supports two HDMI 2.0 interfaces each one capable of driving 4K images. If both HDMI outputs are used then each can be driven up to 4Kp30, however if only HDMI0 interface is being used then images up to 4Kp60 are possible.
One of the interfaces is configured as Standard A socket.

1.6. GPIO Bank (Raspberry Pi HAT connector)

There are 28 pins available for general purpose I/O (GPIO), which correspond to the GPIO pins on the Raspberry Pi 4, Model B 40-pin header. These pins have access to internal peripherals; I2C, PWM, SPI, and UART. The BCM2711 ARM Peripherals book describes these features in detail, and the multiplexing options available.

2. Raspberry Pi Compute Module 4

Raspberry Pi Compute Module 4 harnesses the compute power of the popular Raspberry Pi 4 Model B, bringing it to a smaller form factor suitable for integration into products.

Key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K, hardware video decode at up to 4Kp60, up to 8GB of RAM, Gigabit Ethernet, USB 2.0, dual camera interfaces, and PCIe Gen 2 x1 interface.

The optional dual-band 2.4/5.0GHz wireless LAN and 2.4 Ghz radio have modular compliance certification. This allows the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market. Either the onboard antenna or an external antenna kit can be used.

Compute Module 4 has optional onboard eMMC of 8GB, 16GB or 32GB.

2.1. Communication with Raspberry Pi Compute Module

The module supplied offers several interfaces:

- SSH service on port 22
- MQTT-Broker, default at port 1883
- HTTP RESTful API, default at port 8080
- CoAP RESTful API, default at port 5683
- TCP-Server with TSB data format, default at port 3001
- Configuration webserver, default at port 80

All these services (except for ssh) run as docker containers and can be activated or deactivated as required.

2.1.1. ssh login

The default username is “*pi*” and password “*raspberry*”.
Please change this immediately for security reasons!

3. SAME54 co-processor

The Microchip SAME54 co-processor can be addressed in several ways. Communication takes place in the 3.4. TSB (Taskit Serial Bus) Packet data format. It is equipped with an encryption-capable bootloader and can be supplied with updates.

3.1. CoAP API

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks on the Internet of Things. Like HTTP, CoAP is based on the wildly successful REST model: Servers make resources available under a URL, and clients access these resources using methods such as GET, PUT, POST, and DELETE. The SAME54 CoAP API support GET and PUT methods.

URL	Methods	Description
/serial number	GET	Return 23 Byte long base64 encoded unique serial number
/version	GET	Version number
/systime	GET/PUT	64 bit unsigned int board time in unix time format/
/restart	PUT	Restart the board
/reset_configuration	PUT	Resets the board to default states
/save_configuration	PUT	Save current board configuration in flash
/reset	PUT	Resets the ADC to the default states
/readdata	GET	Get the current
/readreg/#	GET	Read register address # (see Register description) in hexadecimal, exp.: 0027 to read version number
/writereg/#	PUT	Write register address # (see Register description) in hexadecimal, exp.: 0030 to restart the board

Table 2. CoAP resources

3.2. AT Command API

The AT command API can be reached with TSB type ‘text’ and TSB channel 0. A command must begin with 'at' and end with a carriage return and / or new line. Every command is answered with 'OK' or 'ERROR'.
Exp.: ati\n

Command	Description
ati	Get board identifiyer
atv	Get software version
athelp	Get help for at commands
atsave	Save current configuration on eeprom
atstats	Get statistics
atstatreset	Reset statistics
atrreg=rrrr	Read 16Bit register (see Register description); rrrr = register address in hexadecimal, exp.: 0027 to read version number
atwreg=rrrrvvvv	Write 16 bit register (see Register description) rrrr = 16 bit register address in hexadecimal, exp.: 0028 vvvv = 16 bit value in hexadecimal, exp.: 1a2b

Table 3. AT Commands

3.3. Register description

Name	Address (hex)	R/W	Description
Serial number		R	Board serial number
Version		R	Software version
Systime		R/W	Onboard real time clock
Restart board		W	Restart Board
Reset configuration		W	Factory default reset
Save configuration		W	Save current configuration

Table 4. Register description

3.4. TSB (Taskit Serial Bus) Packet

The TSB Packet is a checksum-secured packet format especially for serial data.
It is COBS (Consistent Overhead Byte Stuffing) coded and provided with destination/source and type information.

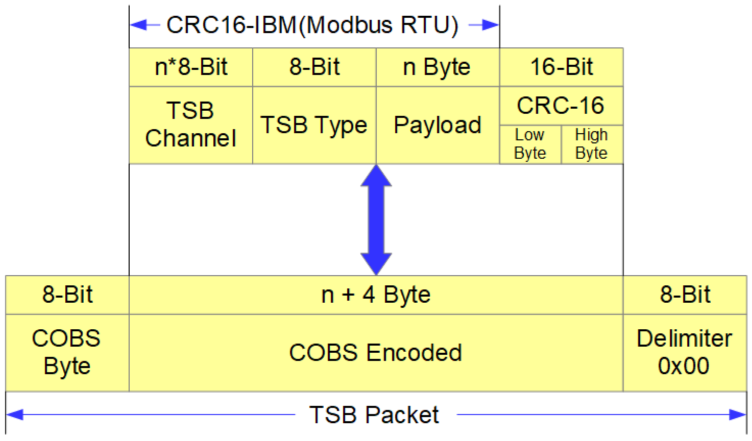

Figure 1. TSB Packet structure

3.4.1. TSB channel

The TSB channel represents the destination or source of a TSB packet. A distinction is made between areas. Channel 0 stands for the board itself, under which the special properties such as serial number, time, reset states and much more can be addressed and configured.

channel	definition	value (hex)	value (dec)	source / destination
base		0x00	0	the board itself
port channel	port 1 port 2 … port 127	0x01 - 0x7F	1 – 127	IO ports by number
routing	port 0.x port 1.x ... port 127.x	0x80 - 0xFF	128 - 256	IO ports beyond the first board in line, e.g. 1.0 is the next board that is connected to the 1st IO port. (See TSB Routing)

Table 5. TSB channel

3.4.2. TSB types

Type	Value (hex)	Value (dec)	Interpretition
raw	0x01	1	Raw data
text	0x02	2	TSB type for AT command API
bline	0x	17	Taskit Beaconline data
bline_v2	0x	18	Taskit Beaconline version 2 data
cwb	0x	19	Taskit corona warn beacon data
hci	0x	21	radio HCI data
coap	0x	33	TSB type for CoAP API
json	0x	48	Data in json format
cbor	0x	49	Data in cbor format
can	0x	65	Data for CAN API
senml_json	0x	110	Senml data in json format
sensml_json	0x	111	Sensml data in json format
senml_cbor	0x	112	Senml data in cbor format
sensml_cbor	0x	113	Sensml data in cbor format
senml_exi	0x	114	Senml data in exi format
sensml_exi	0x	115	Sensml data in exi format
influx	0x	117	Data in line protocol format
error	0x7F	127	TSB type for error messages
test	0x	202	TSB type for testdata

Table 6. TSB types

3.4.3. CRC-16

The checksum is built over the TSB channel byte(s) + TSB type byte + payload. It is calculated with CRC16-IBM (modbus), with the start value 0xFFFF, poly 0x8005, LSB.

3.4.4. COBS coding

Consistent Overhead Byte Stuffing (**COBS**) is an algorithm for encoding data bytes that results in efficient, reliable, unambiguous packet framing regardless of packet content, thus making it easy for receiving applications to recover from malformed packets.
It employs a particular byte value, typically zero, to serve as a *packet delimiter* (a special value that indicates the boundary between packets). The algorithm replaces each zero data byte with a non-zero value so that no zero data bytes will appear in the packet and thus be misinterpreted as packet boundaries.

Byte stuffing is a process that transforms a sequence of data bytes that may contain 'illegal' or 'reserved' values (such as packet delimiter) into a potentially longer sequence that contains no occurrences of those values. The extra length of the transformed sequence is typically referred to as the overhead of the algorithm. The COBS algorithm tightly bounds the worst-case overhead, limiting it to a minimum of one byte and a maximum of ⌈*n*/254⌉ bytes (one byte in 254, rounded up). Consequently, the time to transmit the encoded byte sequence is highly predictable, which makes COBS useful for real-time applications in which jitter may be problematic. The algorithm is computationally inexpensive, and its average overhead is low compared to other unambiguous framing algorithms.
The COBS is coded over the TSB channel byte(s) + TSB type byte + payload + CRC16 bytes.

3.4.5. Routing

If the channel is greater than 127 (the most significant bit in the channel byte is '1'), the following byte does not represent the type, but the next channel hop.

	COBS	Channel hop1	Channel hop2	Channel hop n (if Channel hop n-1 > 0x7F)	TSP type	Payload	CRC	Delimiter
Bytes	1	1	1	n	1	n	2	1
Exp.:	0x0A	0x83	0x01	-	0x02	Ati\r	0x8D9B	0x00

Table 7. TSB Routing

3.5. MQTT

MQTT is a standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth.

3.5.1 MQTT topics

MQTT is used for demultiplexing. The source, the target, the TSB channel and the TSB type can be found in the topic information.

The first part of the topic can be freely selected as an option in the taskit tsbmqtt software and can serve as an alias, e.g. 'voltage1'.

The second part of the topic represents the direction. 't' for transmit to the module. 'r' for receive, from the module.

The third part of the topic stands for the TSB type and can be found in table x. TSB types can be found at TSB types.

The fourth part of the topic represents the TSB channel (TSB channel). The routing works via a point separation of the hops, e.g. 1.0 for the base channel from the second gpio module that is connected to port 1 on the first module (TSB_Routing).

Alias	Direction	TSB type	TSB channel
alias	t or r	TSB types	TSB channel

Table 8. MQTT topics

Exp.: publish 'ati\r' to gpio/t/text/0 to get the module information as response on gpio/r/text/0.

4. Start up with gpio boards

4.1. Connect the gpio board

Connect the gpio board to any USB host port with a Micro USB A cable. The green LED1 should turn on and off every 1 second. A new device should appear in the Windows device manager or in the Linux /dev directory.

4.2. taskit Software

Depending on which interface you prefer, a different compilation of software can be started. Taskit offers software versions for Microsoft Windows, Mac and Linux x86 or Linux ARM. All programs are command line tools.

4.2.1. serialnet

serial net establishes a connection between different media. So serial net can multiplex a serial interface over a TCP server.
The connection type for first device is specified with the command (client, device or server) and the connection type for the second device is specified by the parameter.
Multiple devices can connect to the server and will receive all data from the serial connection. All Data sent to the server will be sent to the serial connection.
Each side can have the following connection types:

serial - opens an UART or an USB Port
client - connects to a tcp server
server - instantiates a tcp server and listens for tcp clients
stdio - connects to stdin and stdout of the console

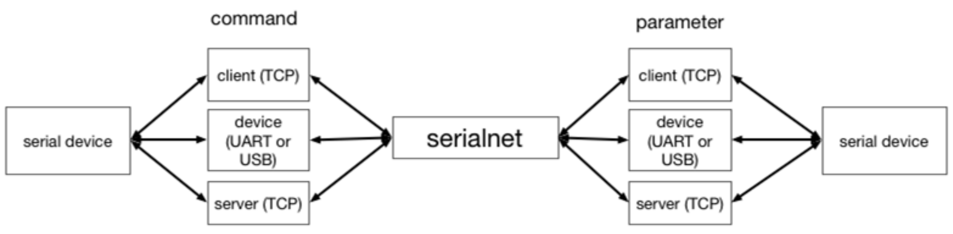

Figure 2. serialnet: connection concept

Command	Description
client	0.00 €
config	writes the config file: 'serialnet.yaml' with all available flags in the current directory
help	help about any command
serial	serial device tty or com (1) and serial or client or server mode on the other side (2)
server	serial data as tcp server (1) and serial or client or server mode on the other side (2)
stdio	serial stdio (1) and serial or client or server mode on the other side (2)

Table 9. serialnet: commands

Flag	Description
--baud1 int	baudrate (default 115200)
--baud2 int	baudrate (default 115200)
--cfgpath	path of configfile (default ".")
-h, --help	help about command
--port1 int	TCP-Port (default 3000)
--port2 int	TCP-Port (default 3000)
--ser1 string	Serial Device Name (default "/dev/ttys0")
--ser2 string	Serial Device Name (default "/dev/ttys0")
--tsbCh1 string	tsb channel, empty string means no tsb mode (default)
--tsbCh2 string	tsb channel, empty string means no tsb mode (default)
--tsbTyp1 string	tsb typ (default "raw"), only necessary in tsb mode
--tsbTyp2 string	tsb typ (default "raw"), only necessary in tsb mode
--url1 string	URL of TCP-Server (default "localhost")
--url2 string	URL of TCP-Server (default "localhost")
--verbose1	verbose outputs
--verbose2	verbose outputs

Table 10. serialnet: flags

Example:
To connect to a serial device with a baudrate of 115200 and a TCP server on port 5000:
serialnet --ser1 COM1 --baud1 115200 --port2 5000 serial server

To connect another serial device with the first example:
serialnet –ser2 COM2 –port1 5000 client serial

4.2.2. tsbcoap

Connect to a TSB server(serialnet) and creates a CoAP server and a HTTP server. Received CoAP or HTTP requests will be packaged into TSB and sent to the TSB server and the other way around.

Flag	Description
--cfgpath string	Path of configfile (default ".")
--cport int	CoAp server port (default 5683)
--hport int	HTTP server port (default 8080)
--tport int	Port of TSB Server (default 3001)
--turl string	Url from TSB Server (default "localhost")
--verbose	verbose outputs

Table 11. tsbcoap: flags

Example:
To connect to serialnet on port 5000:
tsbcoap –tport 5000

Send HTTP request to get serial number of a local device via curl:
curl -X GET localhost:8080/serialnumber

Reset a device at a known ip address via curl:
curl -X PUT 192.168.0.123:8080/reset

Get and put register value via curl:
curl -X GET localhost:8080/readregister/0100
curl -X PUT -d "1A8F" localhost:8080/writeregister/0100

4.2.3. tsbmqtt

Command	Description
client	connects to a tsb client
config	writes the config file: 'tsbmqtt.yaml' with all available flags in the current directory
help	help about any command
device	connects to a serial device
server	instantiates a tsb server

Table 12. tsbmqtt: commands

Flag	Description
--baud int	baudrate (default 115200)
--cfgpath string	path of configfile (default ".")
--clientid string	ClientID
--dev string	Serial Device Name (default "/dev/ttys0")
-h, --help	help for tsbmqtt
--mport int	Port to use to connect to the MQTT broker (default 1883)
--murl string	Url from MQTT broker (default "localhost")
--passwd string	Password from MQTT broker
--persist	persist mode
--pub string	publish topic (default "r")
--qos int	Quality of service 0-2 (default 1)
--reverse	changes pub and sub topic
--sub string	subscribe topic (default "t")
--timestamp	puts timestamp in the topic
--topic string	The mqtt topic prefix. (default "m2go")
--tport int	The mqtt topic prefix. (default "m2go")
–turl string	Url from tsb Server (default "localhost")
--user string	User name
--verbose	verbose outputs

Table 13. tsbmqtt: flags

Example:
To connect to serialnet on port 5000:
tsbmqtt –tport 5000

4.2.4. tsbweb

Creates a Webserver and hosts a website that can be specified. It also connects to a TSB server(serialnet) or directly to a serial TSB device.
It also opens multiple Websockets with different functionalities like transmitting CoAP messages or TSB converter.

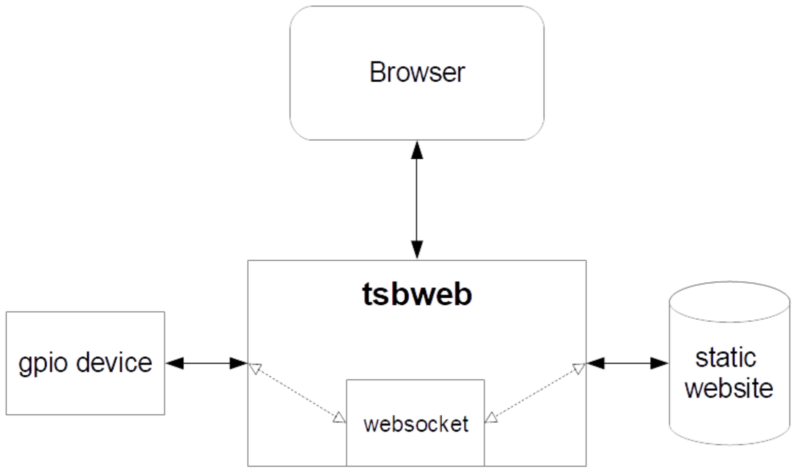

Figure 3. tsbweb: connection concept

Command	Description
config	writes the config file: 'tsbweb.yaml' with all available flags in the current directory
help	help about any command
device	connects to a serial device
tcp	TSB tcp client

Table 14. tsbweb: commands

Flag	Description
--cfgpath string	Path of configfile (default ".")
--baud int	Baudrate (default 115200)
--dev string	Serial Device Name (default "/dev/ttys0")
-h, --help	help for tsbweb
--hport int	Port of http (default 80)
--keypath string	Path of tls keys: cert.pem and key.pem (default ".")
--tls	http with tls (https)
--tport int	Port of TSB Server (default 3000)
--turl string	Url from TSB Server (default "localhost")
--verbose	verbose outputs
--www string	Path of website (default "./www")

Table 15. tsbweb: flags