RA6W1 Host Interface

The RA6W1 is a highly integrated ultra-low-power Wi-Fi MCU that allows developing a complete Wi-Fi solution on a single chip. The RA6W1 provides support for various peripheral communication interfaces, including SPI, SDIO, and UART.

Contents

Figures

Tables

1. Terms and Definitions

AP	Access Point
BSP	Board Support Package
CMD	Command
COM	Communication Port
EVB	Evaluation Board
GPIO	General Purpose Input Output
PPA	Programmable Pin Assignment
RX	Receive
SDIO	Secure Digital Input Output
SPI	Serial Peripheral Interface
TX	Transmit
UART	Universal Asynchronous Receiver
WAP	Wireless Application Protocol

AP	Access Point
BSP	Board Support Package
CMD	Command
COM	Communication Port
EVB	Evaluation Board
GPIO	General Purpose Input Output
PPA	Programmable Pin Assignment
RX	Receive
SDIO	Secure Digital Input Output
SPI	Serial Peripheral Interface
TX	Transmit
UART	Universal Asynchronous Receiver
WAP	Wireless Application Protocol

2. References

[1] RA6W1, Datasheet, Renesas Electronics.
[2] RA6W1, DEVKT Electric Schematic, Renesas Electronics.
[3] Evaluation Kit for RA6M4 Microcontroller Group EK-RA6M4, Quick Start Guide, Renesas Electronics.

Note 1 References are for the latest published version, unless otherwise indicated.

3. Introduction

This document describes how an external processor system (referred to as External Host) communicates with the RA6W1 through UART, SPI, and SDIO physical interface protocols PMGR_LLD_POWER_MODE_SLEEP3.

4. UART Interface

The RA6W1 UART interface provides an industry compliant serial interface for communicating with other devices. Three independently configurable UARTs are available with the support of RS-232 and RS-485 protocols. The UART pins can be assigned to any of the unused GPIO pins through the Programmable Pin Assignment (PPA) function. The pin definitions for the UART interfaces are explained in Ref. [1].

- UART0 is used as Wi-Fi UART (CLI) and is assigned to the GPIO pairs [P0_00(TX):P0_01(RX)].
- UART1 is used as AT UART (AT Command) and is assigned to the GPIO pairs [P0_04(TX):P0_05(RX)].

4.1 UART Pin MUX Configuration

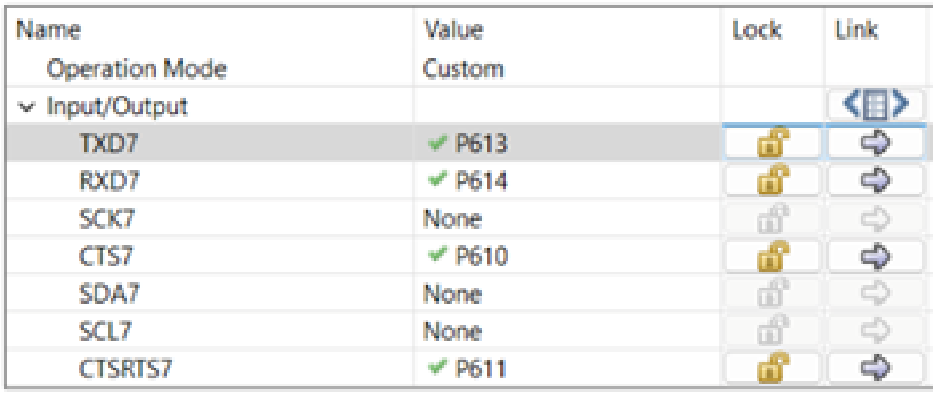
The UART interface serves as the physical layer for transmitting and receiving data between the external MCU and the RA6W1. AT commands communication is typically carried out over this UART interface. The MCU sends AT commands to the RA6W1 through the UART TX pin, and the MCU receives the response through the UART RX pin.

Table 1. Pin MUX configuration for AT command – UART

Usage	RA6Wx EVB
RXD	P0_04
TXD	P0_05
RTS	P0_08
CTS	P0_09
GND	J219-P5 (GND)

4.2 RA6M4 + RA6W1 AT Command over UART Example

An RA6M4 example program for AT command over UART can be downloaded from the RA6W1 example repository.



Name	Value	Lock	Link
Operation Mode	Custom		
Input/Output			
TXD7	✓ P613		
RXD7	✓ P614		
SCK7	None		
CTS7	✓ P610		
SDA7	None		
SCL7	None		
CTSRTS7	✓ P611		

Figure 1. UART pins from RA6M4 program configuration

Table 2 shows pin configuration for AT command interfacing through UART of EK-RA6M4 (MCU) with the RA6W1.

Table 2. UART pin connection for interfacing with EK-RA6M4

RA6Wx EVB	EK-RA6M4
P0_04 (RX)	P613 (TX)
P0_05 (TX)	P614 (RX)
P0_08 (RTS)	P610 (CTS)
P0_09 (CTS)	P611 (RTS)
J219-P5 (GND)	GND

The following jumper connections are also necessary as shown in Figure 2

- Connect FD3_CS on J3 to FD3_CS_D on J3.
- Connect FD2_DO on J3 to FD2_DO_D on J3.

- Connect FD1_DI on J3 to FD1_DI_D on J3.
- Connect FD0_SCLK on J3 to FD_SCLK_D on J3.

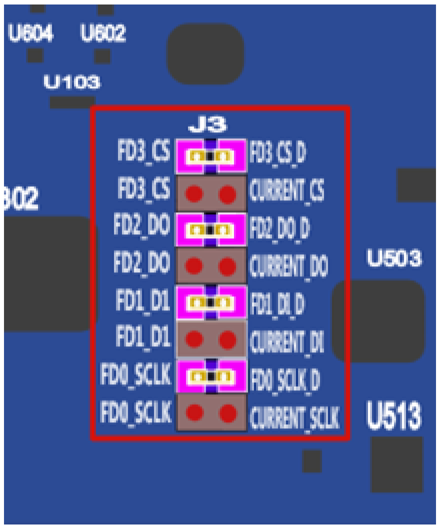


Figure 2. Jumper connections for AT command

NOTE
Any ground pins on the EK-RA6M4 can be used.

This program works by taking AT commands as user input through the RA6M4's USB console and communicates with the RA6W1 through UART, and then the response is displayed on the USB console of RA6M4. To understand how to load a program on EK-RA6M4, see Ref. [3].

5. SPI Interface

The Serial Peripheral Interface (SPI) protocol is a synchronous serial communication interface used to transfer data between microcontrollers and peripheral devices. This interface allows high-speed data transfer and is typically used for short-distance communication. SPI operates with a master-slave architecture, where the master device controls the clock signal (SCK) and data transmission, and uses four primary signals:

- MISO (Master In Slave Out) – Carries data from slave to master.
- MOSI (Master Out Slave In) – Carries data from master to slave.
- SCK (Serial Clock) – Provides the clock signal from the master.
- SS (Slave Select) – Indicates which slave device is active.

SPI is simple, fast, and widely used in embedded systems, offering full-duplex communication where data can be transmitted and received simultaneously.

5.1 SPI Protocol – Message Format

The format of the messages sent/received to/from the external processor is the RA6W1 protocol format over SPI physical interface. Figure 3 shows the message format and parameters included in the RA6W1.

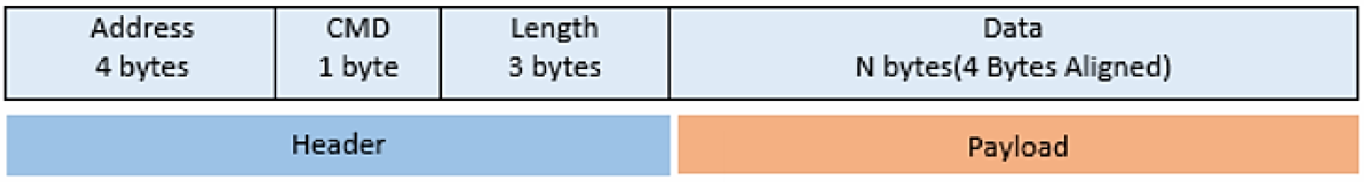


Figure 3. Basic SPI message format

Table 3 shows the address list used by External Host.

Table 3. SPI address list

Address type	Address
AT Command	0x50080260
Response command	0x50080258
Buffer address	Received from slave in response message

Table 4 shows the format of CMD fields.

Table 4. SPI command format

Bit	Field	Description
7	Auto_Inc	1: Internal Address auto-increment 0: Address Fixed (Not used)
6	Read/Write	1: Read 0: Write
5:2		Not Used
1:0	CHIP_ID[1:0]	00: CHIP #0 (Default)

Length is a payload length of the data field.

5.2 General Command (Write request) – Sequences and Structures

Host to Slave write operations are performed in three SPI transactions as shown in Figure 4.

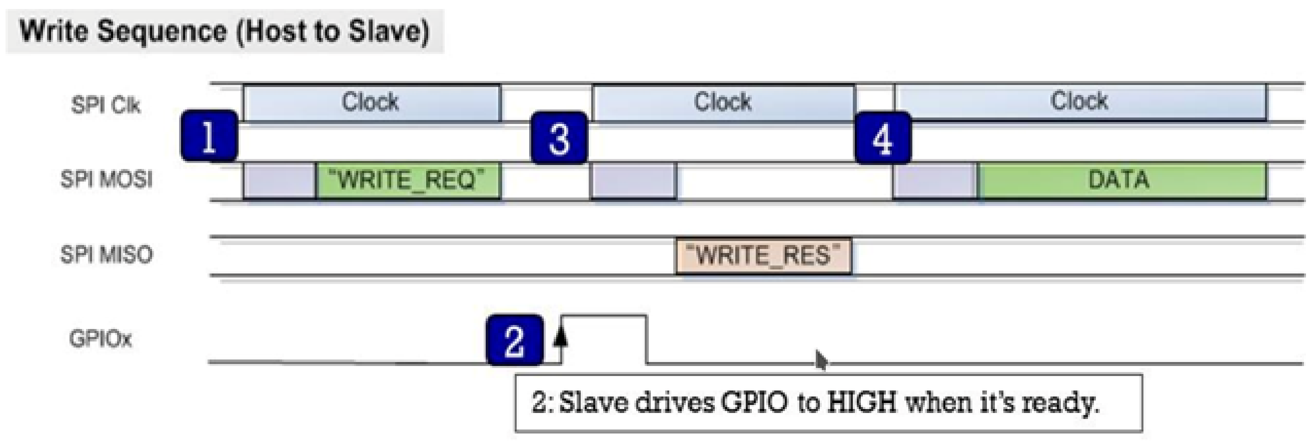

Write Sequence (Host to Slave)
SPI Clk
SPI MOSI
SPI MISO
GPIOx
Clock
Clock
Clock
1
3
4
2
"WRITE_REQ"
DATA
"WRITE_RES"
2: Slave drives GPIO to HIGH when it's ready.

Figure 4. AT command sequence

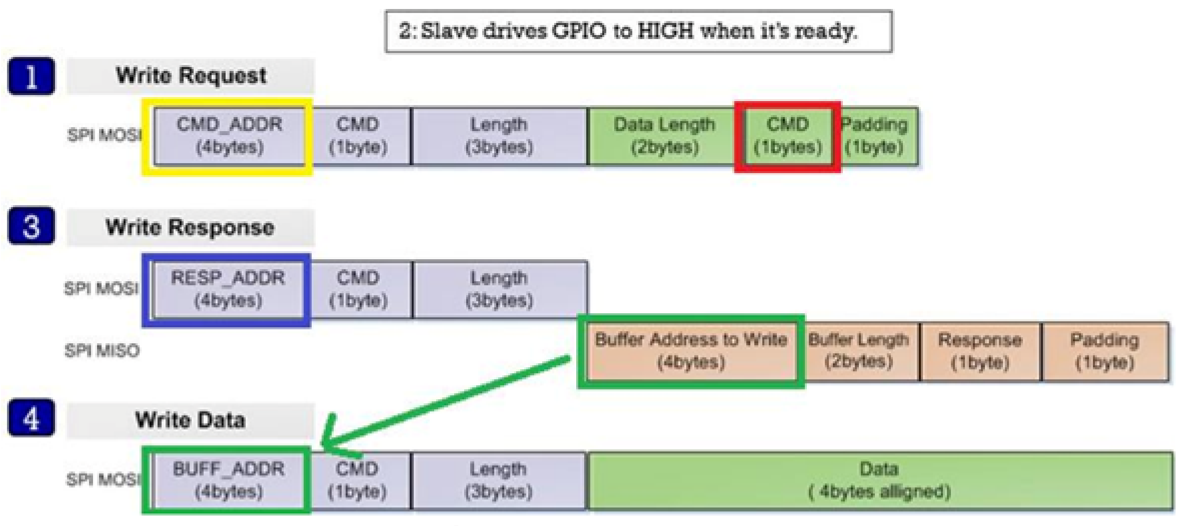


Figure 5. Structure of AT command

1: The Host sends a WRITE_REQ command (0x80, red rectangle in Figure 5) to the General Command address (0x50080254) (yellow rectangle in Figure 5).

2: The Host should wait for GPIO interrupt line is High from slave.

3: The Host reads the Write Response message by Response Command address (0x50080258, blue rectangle in Figure 5) and parses it.

4: The Host sends data to address (BUFF_ ADDR) which is received from the Slave in the Write Response message (green rectangle in Figure 5).

Example:

When the host wants to write 8-byte data (0x8877665544332211) to the RA6W1:

- The Host sends (0x50-0x08-0x02-0x54)-(0x80)-(0x00-0x00-0x04)-(0x08-0x00-0x80-0x00).
- The Host waits until GPIO interrupt line is high from theRA6W1.
- The Host sends (0x50-0x08-0x02-0x58)-(0xC0)-(0x00-0x00-0x08), then reads responses from theRA6W1. If the buffer address from Slave is 0x12345678 for easy description, the read data should be 0x78-0x56-0x34-0x12-

0x08-0x00-0x81-0x00.

- The Host sends (0x12-0x34-0x56-0x78)-(0x80)-(0x00-0x00-0x08)-(0x11-0x22-0x33-0x44-0x55-0x66-0x77- 0x88).

NOTE
The payload data is transmitted MSB first and little-big endian system.

NOTE
An interval of at least a hundred microseconds is required between each AT command request. If the interval between the two requests is too short, there is a possibility that two Interrupt Events are recognized as one. The interval depends on the type of application or CPU load. Roughly, when the CPU clock is 160 MHz, an interval of around 100 μs is required.

5.3 AT Command – Sequences and Structures

AT commands are instructions used to control a modem. Every command line starts with AT or at. Start AT is the prefix that informs the modem about the start of a command line. It is not part of the AT command name.

Figure 6 shows how to use the AT command through SPI on theRA6W1. This is because AT command uses a pre-determined address, and the maximum size of data is defined.

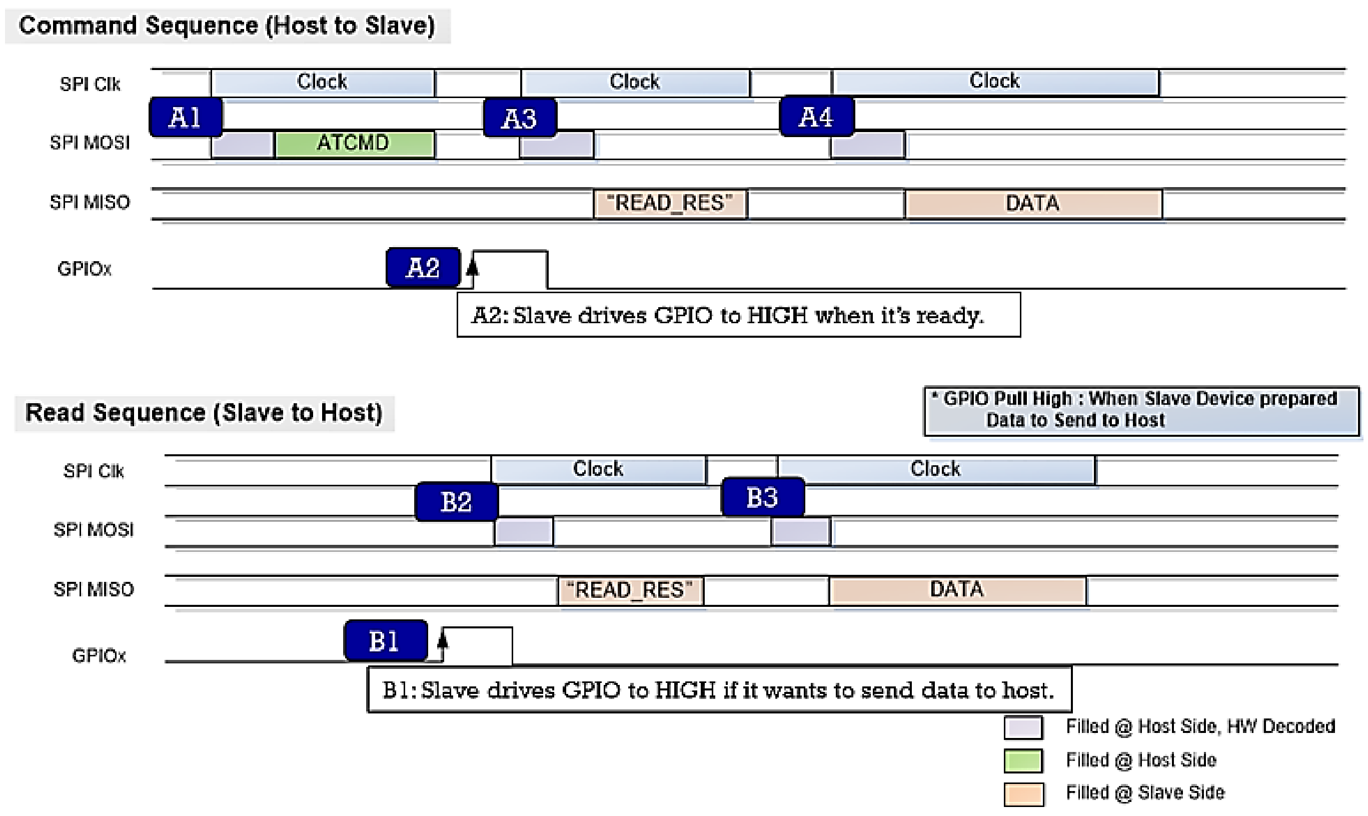

Command Sequence (Host to Slave)
SPI Clk
Clock
Clock
Clock
A1
A3
A4
SPI MOSI
ATCMD
SPI MISO
"READ_RES"
DATA
GPIOx
A2
A2: Slave drives GPIO to HIGH when it's ready.
Read Sequence (Slave to Host)
* GPIO Pull High : When Slave Device prepared Data to Send to Host
SPI Clk
Clock
Clock
B2
B3
SPI MOSI
SPI MISO
"READ_RES"
DATA
GPIOx
B1
B1: Slave drives GPIO to HIGH if it wants to send data to host.
Filled @ Host Side, HW Decoded
Filled @ Host Side
Filled @ Slave Side

Figure 6. AT command sequence

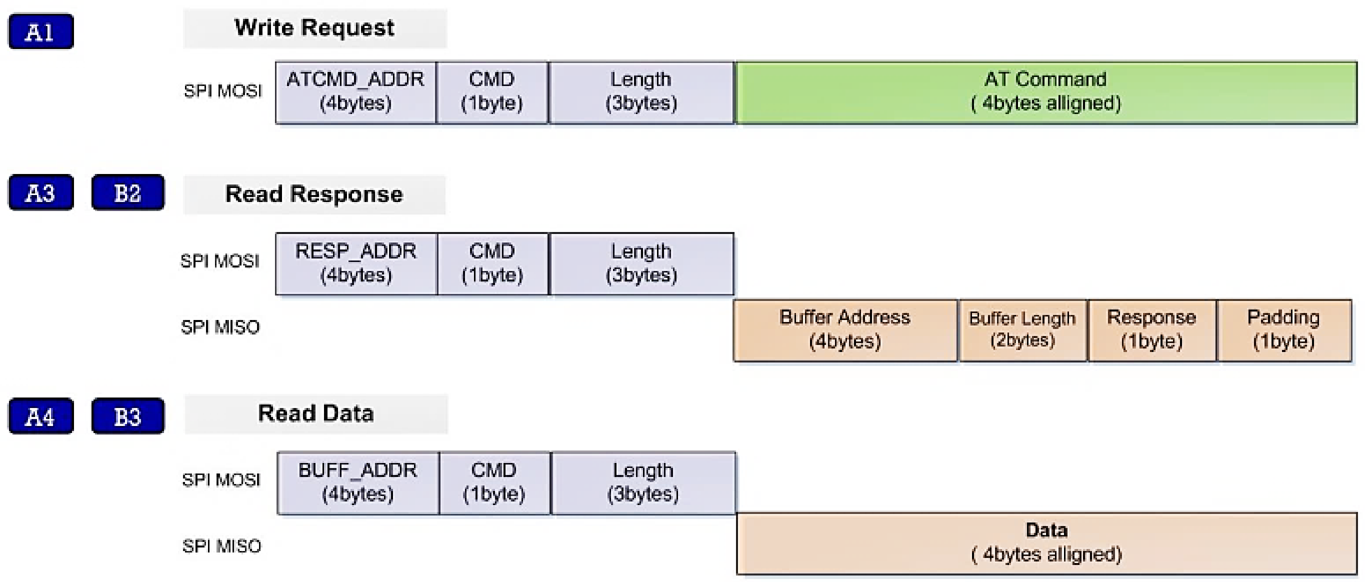


Figure 7. Structure of AT command

A1: The host sends an AT command to AT command address.

A2: The host waits for GPIO interrupt line to go high.

A3: The host reads the response message from address and parses it using struct st_spi_dev_resp.

A4: The host reads OK/Error, or data from address (BUF_ADDR), depending on the command type.

Example:

- To write `AT+VER` command to the RA6W1, the host sends:
  (0x50-0x08-0x02-0x60)-(0x80)-(0x00-0x00-0x08)-('A'-'T'-'+'-'V'-'E'-'R'-0x00-0x00)
- The read sequence after writing is the same as the following examples of B1~B3.

NOTE
The payload data is transmitted MSB first and little-big endian system (see Figure 11).

B1: The slave toggles the interrupt line to go high to inform the host when it wants to send data to the host.

B2: The host reads the response message from Response Command address, and parses it using struct `st_spi_dev_resp`

B3: The host reads data from address (BUF_ADDR) parsed from the response message.

5.4 ESC Command – Sequences and Structures

Figure 8 shows how to use the ESC command through SPI on the RA6W1. This is because the ESC command uses a predetermined address, and the maximum size of data is defined. Figure 9 shows the structure of ESC command.

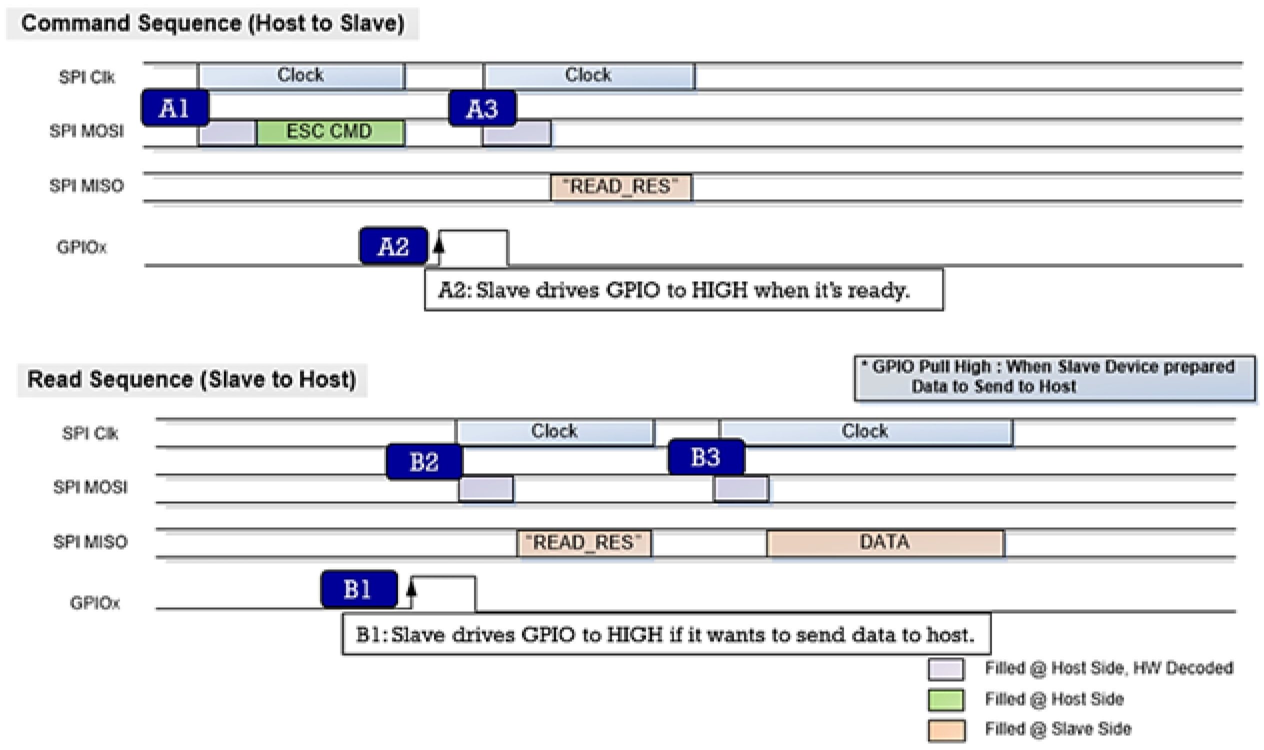

Command Sequence (Host to Slave)
SPI Clk
SPI MOSI
SPI MISO
GPIOx
Clock
Clock
A1
ESC CMD
A3
"READ_RES"
A2
A2: Slave drives GPIO to HIGH when it's ready.
Read Sequence (Slave to Host)
* GPIO Pull High : When Slave Device prepared Data to Send to Host
SPI Clk
SPI MOSI
SPI MISO
GPIOx
Clock
Clock
B2
B3
"READ_RES"
DATA
B1
B1: Slave drives GPIO to HIGH if it wants to send data to host.
Filled @ Host Side, HW Decoded
Filled @ Host Side
Filled @ Slave Side

Figure 8. ESC command sequence

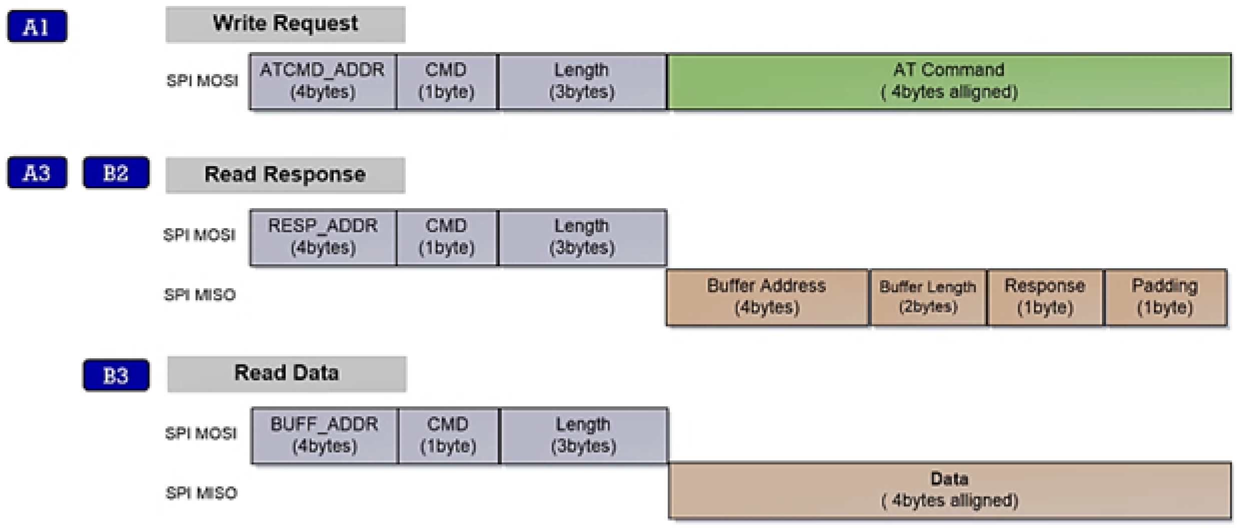


Figure 9. Structure of ESC command

A1: The host sends the ESC command to the AT command address.

A2: The host waits for GPIO interrupt line to go high.

A3: The host reads the response message from address and parses it using st_spi_dev_resp. The result for the ESC command is sent to the host as the response field of `st_spi_dev_resp`. The response field is a 1-byte decimal value. The value of 0x20 is a result of OK. All other values are ERROR. And in this case, the value of the buf_address field is read as 0xffffffff, and the value of the host_length field is read as 0x0. Therefore, the subsequent Read Sequence is not required.

Example:

- To write <ESC>S010,192.168.0.18,43310,abcde12345 command to the RA6W1, the host sends: (0x50-0x08-0x02-0x60)-(0x80)-(0x00-0x00-0x24)-(<ESC>-‘S’-‘0’-‘1’-‘0’-‘,’-‘1’-‘9’-‘2’-‘.’-‘1’-‘6’-‘8’-‘.’-‘0’-‘.’-‘1’-‘8’-‘,’-‘4’-‘3’-‘3’-‘1’-‘0’-‘,’-‘a’-‘b’-‘c’-‘d’-‘e’-‘1’-‘2’-‘3’-‘4’-‘5’-0x00)
- The read sequence after writing is the same as the following examples of B1~B2.

NOTE
The payload data is transmitted MSB first and little-big endian system, see Figure 10.

B1: The slave toggles the interrupt line to go high to inform the host when it wants to send data to the host.

B2: The host reads the response message from Response Command address and parses it using `st_spi_dev_resp`.

B3: The host reads data from address (BUF_ADDR) parsed from the response message.

NOTE
An interval of at least a hundred microseconds is required between each ESC command. If the interval between the two ESC requests is too short, there is a possibility that two Interrupt Events are recognized as one. The interval depends on the type of application or CPU load. Roughly, when the CPU clock is 160 MHz, an interval of around 100 μs is required.

5.5 Header Format

5.5.1 Write Request (Host to Slave)

Figure 10 shows write request (host to slave).

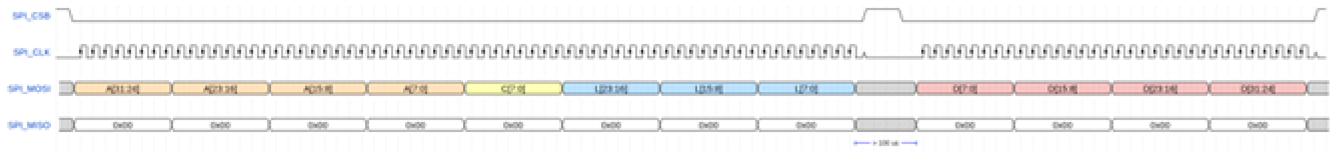


Figure 10. SPI signals for write request

5.5.2 Read Response (Slave to Host)

Figure 11 shows a read response (slave to host).

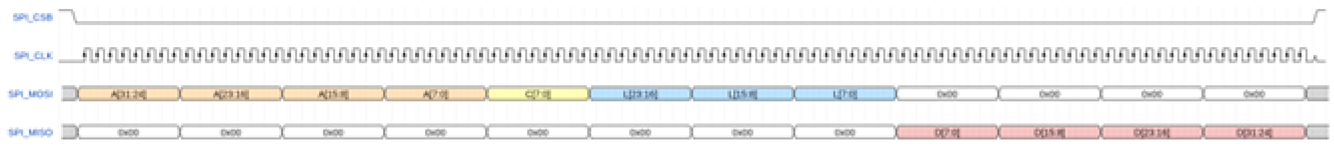


Figure 11. SPI signals for read response

5.6 SPI Pin MUX Configuration

Table 5. Pin MUX configuration – SPI

Usage	RA6Wx EVB
SCLK	P1_00
MOSI	P1_01
MISO	P1_02
CS	P1_03
IRQ	P0_07

5.7 RA6M4 + RA6W1 AT Command over SPI Example

An RA6M4 example program for AT command over SPI can be downloaded from the RA6W1 example repository.

NOTE
Renesas recommends using e^2 studio FSP version 5.2.0 for this program.

5.7.1 Overview

This example works by taking the user input AT command through the console of the RA6M4, bridges it from to SPI of RA6M4 and sends the command to SPI of the RA6W1 and gets the response.

The following are the SPI slave interface pin configurations required in the RA6W1 code for obtaining the SPI connections through the PMOD connector.

```
                BSP_IO_PORT_01_PIN_00, .pin_cfg =
((uint32_t) IOPORT_CFG_DRIVE_STRENGTH_BA_8MA
        | (uint32_t) IOPORT_CFG_PERIPHERAL_PIN
        | (uint32_t) IOPORT_CFG_PORT_DIRECTION_INPUT
        | (uint32_t) IOPORT_CFG_SLEW_RATE_FAST
        | (uint32_t) IOPORT_PERIPHERAL_SPI2_CLK) }, { .pin =
BSP_IO_PORT_01_PIN_01, .pin_cfg =
((uint32_t) IOPORT_CFG_DRIVE_STRENGTH_BA_8MA
        | (uint32_t) IOPORT_CFG_PERIPHERAL_PIN
        | (uint32_t) IOPORT_CFG_PORT_DIRECTION_INPUT
        | (uint32_t) IOPORT_CFG_SLEW_RATE_FAST
        | (uint32_t) IOPORT_PERIPHERAL_SPI2_DI) }, { .pin =
BSP_IO_PORT_01_PIN_02, .pin_cfg =
((uint32_t) IOPORT_CFG_DRIVE_STRENGTH_BA_8MA
        | (uint32_t) IOPORT_CFG_PERIPHERAL_PIN
        | (uint32_t) IOPORT_CFG_PORT_DIRECTION_OUTPUT
        | (uint32_t) IOPORT_CFG_SLEW_RATE_FAST
        | (uint32_t) IOPORT_PERIPHERAL_SPI2_DO) }, { .pin =
BSP_IO_PORT_01_PIN_03, .pin_cfg =
((uint32_t) IOPORT_CFG_DRIVE_STRENGTH_BA_8MA
        | (uint32_t) IOPORT_CFG_PERIPHERAL_PIN
        | (uint32_t) IOPORT_CFG_PORT_DIRECTION_INPUT
        | (uint32_t) IOPORT_CFG_SLEW_RATE_FAST
        | (uint32_t) IOPORT_PERIPHERAL_SPI2_CSN0) }
```

The pins shown inTable 6 should be bridged on the RA6W1 (J203 and J201) to direct the SPI signals to the PMOD connector.

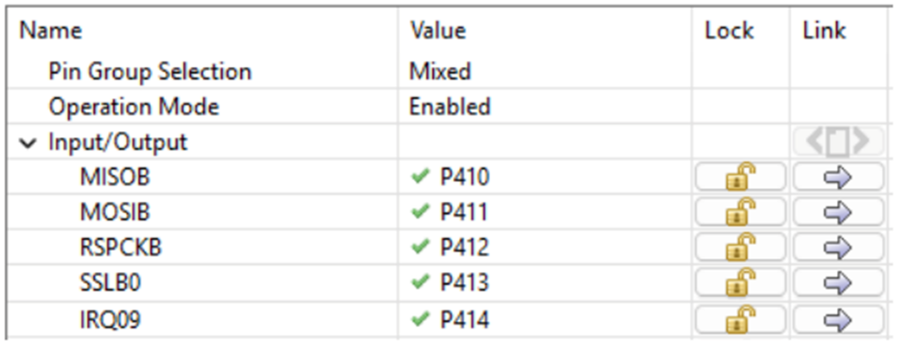
Table 6. RA6W1 EVB jumper configuration for SPI over PMOD

J203	
P1_00	SCLK
P1_01	MOSI
P1_02	MISO
P1_03	CS
J201	
P0_07	INT_1

When the RA6W1 has data to send, it sends an interrupt to notify the master (RA6M4).

NOTE
The jumper connections J3 shown in Figure 2 are not required.

To get the SPI signals over the PMOD2 port of RA6M4, the pins in Figure 11 are used in the program.



Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Enabled		
Input/Output			
MISOB	✓ P410		
MOSIB	✓ P411		
RSPCKB	✓ P412		
SSLB0	✓ P413		
IRQ09	✓ P414		

Figure 12. SPI pins for PMOD2 from RA program configuration

The PMOD connector of the RA6W1 should be connected to the PMOD2 port of the RA6M4 host (Figure 15) for this program to work:

1. Connect the PMOD connector of the RA6W1 to the PMOD2 port of the RA6M4 host (Figure 14) for this program to work.
2. Connect P0_04 to INT_0 in J201 OR connect P0_04 of RA6W1 to P414 of RA6M4.
3. Connect jumpers between P1_00 and FD0_SCLK_DA ,P1_01 and FD1_DI_DA,P1_02 and FD2_D0_DA,p1_03 and FD3_CS_DA. PMOD Connections between J219 and PMOD2 shown in Figure 13.

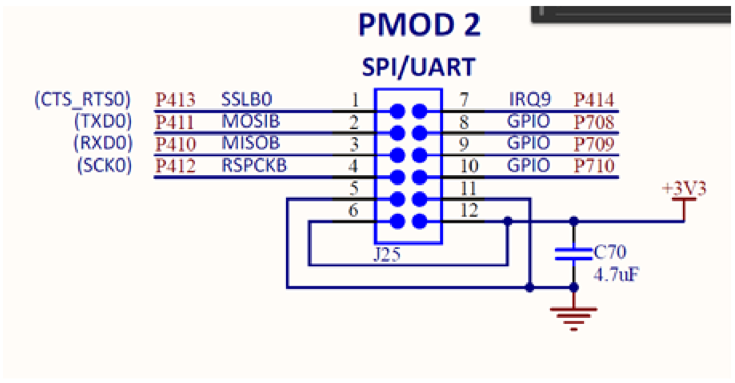


Figure 13. PMOD Connection

4. Connect INT_1 of J219 in REV E EVK to J25-7 (GPO/INT) of PMOD2 in RA6M4.

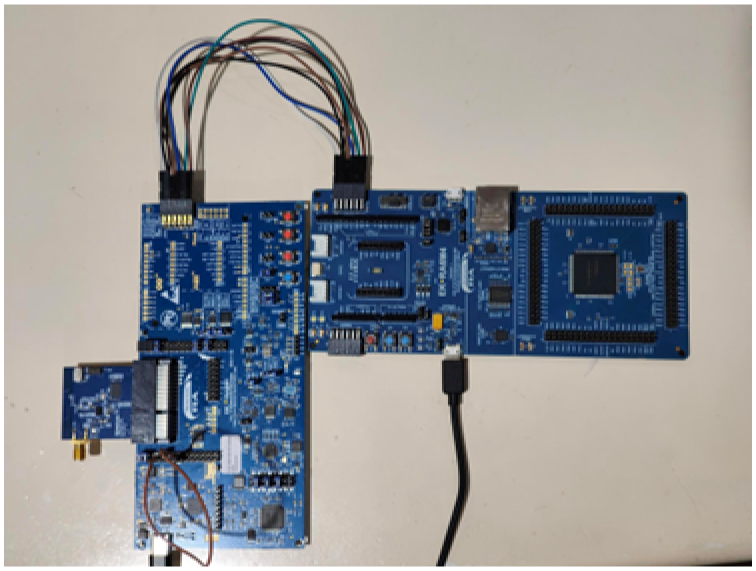


Figure 14. PMOD connection between RA6M4 and RA6W1 through J219

5. You can also set up the SPI interface by connecting the PMOD2 of the RA6M4 and J319 of the RA6W2.
6. When connecting the Pmod (J319) to the RA6M4 through the Pmod interface, the module operates only at 1.8 V. Therefore, ensure that the jumper on J5 is placed between I/O_V1.8V and VCCA_1 to enable proper 1.8 V operation.

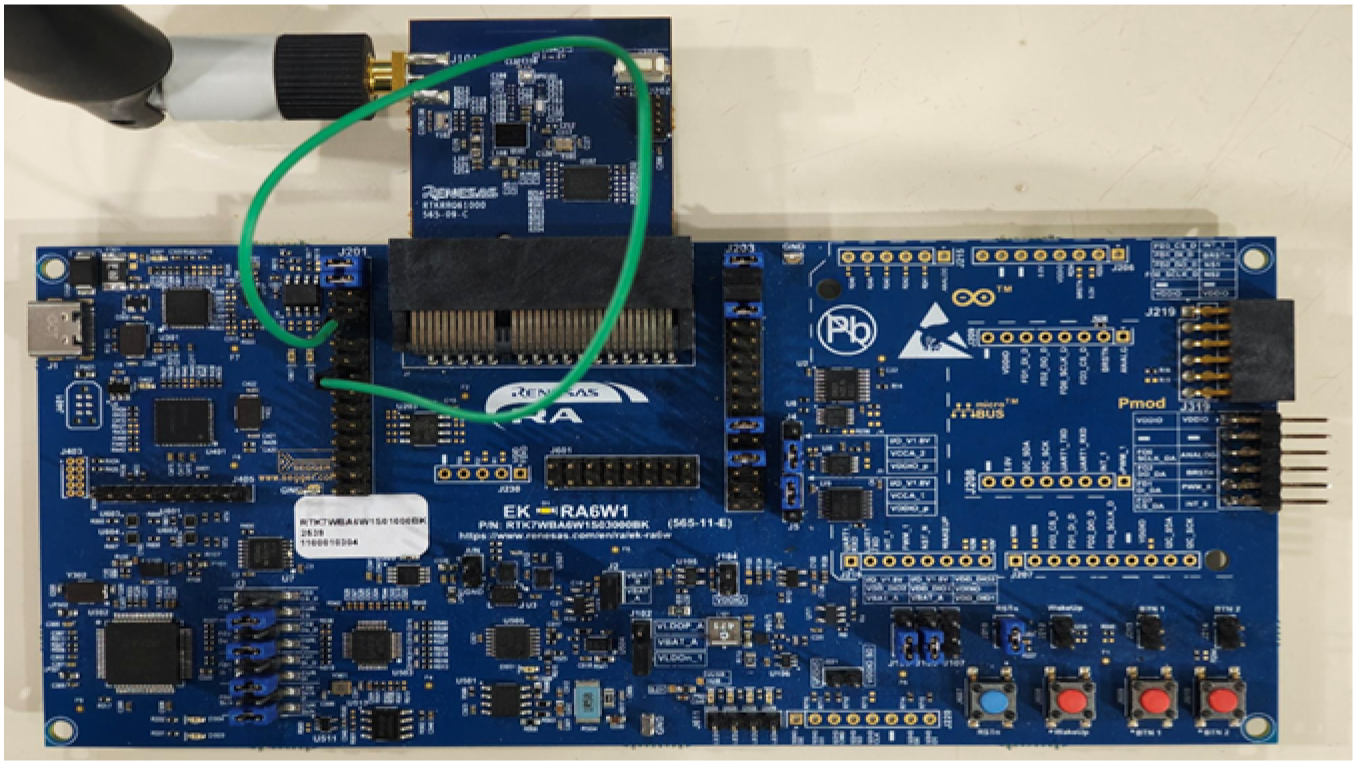


Figure 15. PMOD connection between RA6M4 and RA6W1

NOTE
In case of any limitations on the use of PMOD, the pins mentioned in Table 5 can be used directly.

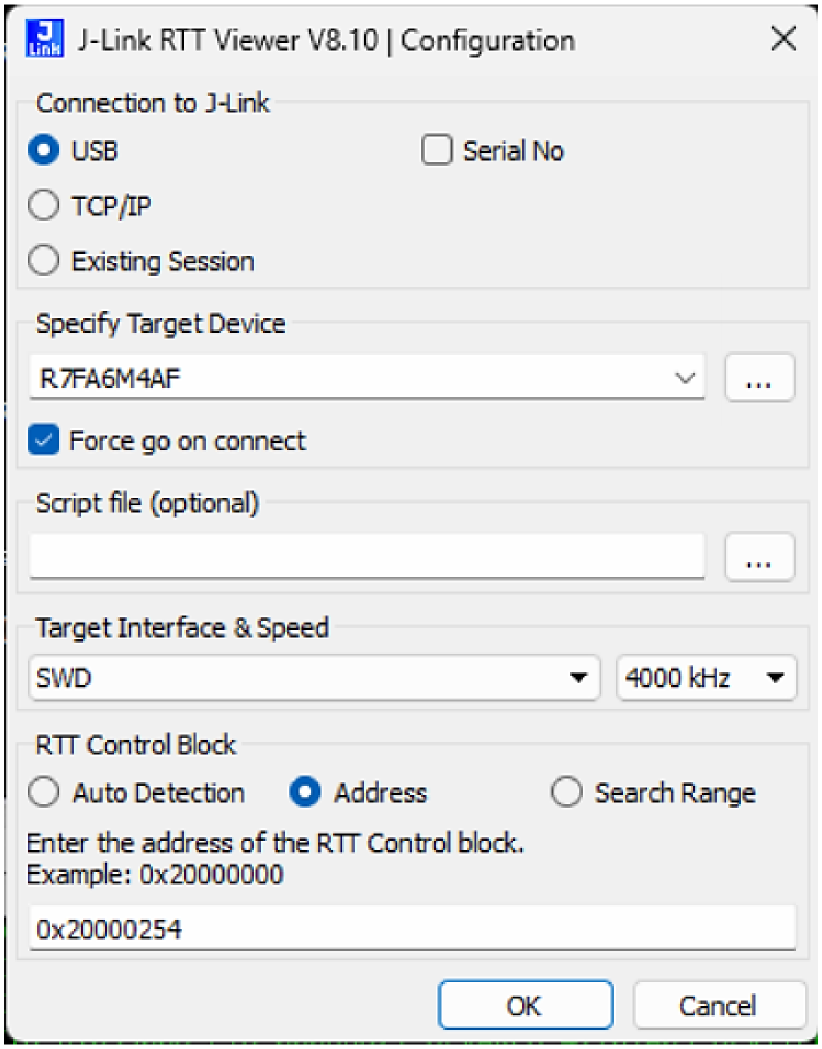
To understand how to load a program on EK-RA6M4, see Ref. [3].

5.7.2 Components

- **RA6M4 USB console** (Type A cable):
  The USB console can be accessed using Tera Term or similar applications. The baud rate setting should be 115200. This console is used to send the AT commands to RA6M4.
- **RA6M4 Debug console** (Type A cable):

The debug port (DEBUG1) of RA6M4 is used for loading images and viewing the debug logs. For this purpose, use J-Link RTT Viewer that can be downloaded from https://www.segger.com/downloads/jlink/.

Figure 16 shows the required J-Link RTT Viewer configuration for the example program.



J-Link RTT Viewer V8.10 | Configuration

Connection to J-Link
- USB [✓]
- TCP/IP
- Existing Session
- Serial No ☐

Specify Target Device: R7FA6M4AF

Force go on connect ✓

Script file (optional):

Target Interface & Speed: SWD, 4000 kHz

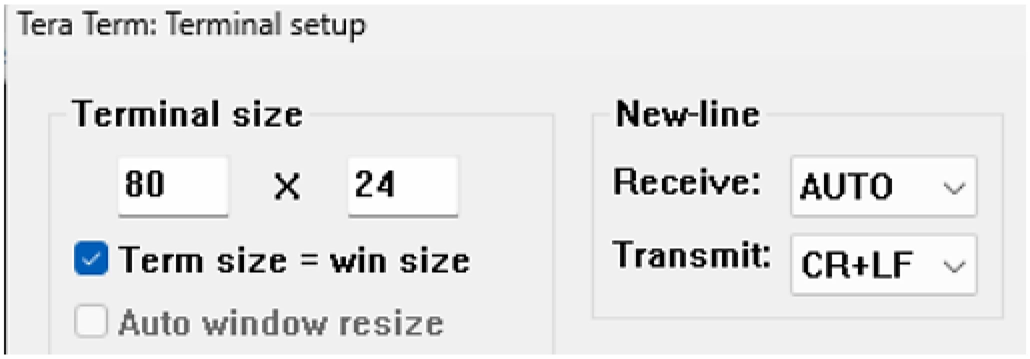
RTT Control Block: Auto Detection [✓ Address] Search Range

Enter the address of the RTT Control block.
Example: 0x20000000

0x20000254

Figure 16. J-Link RTT Viewer configuration

- **RA6W1 CLI console** (Type-C cable):It is used to power the RA6W1 as well as to access the CLI console to view logs. The baud rate should be set to 115200. Also, you need the setting shown in Figure 17 in terminal for New-line receive and transmit.



Tera Term: Terminal setup

Terminal size: 80 x 24

Term size = win size ✓

Auto window resize ☐

New-line

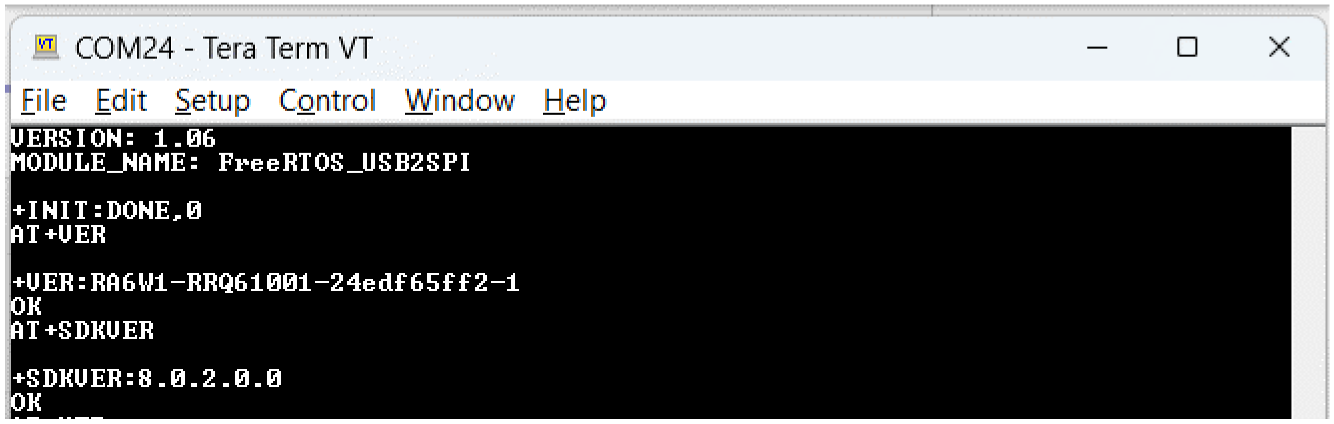
Receive: AUTO

Transmit: CR+LF

Figure 17. Tera Term New-line configuration

5.7.3 Working Example

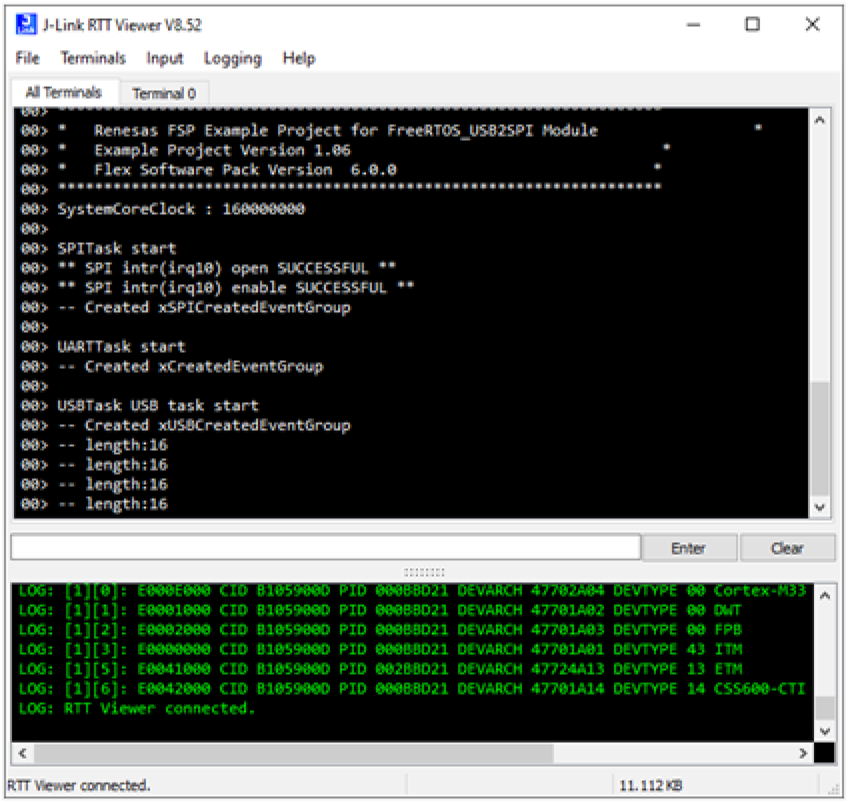
Figure 18 shows sending `AT+VER` and `AT+SDKVER` commands one at a time through the RA6M4 console terminal.



```
VERSION: 1.06
MODULE_NAME: FreeRTOS_USB2SPI

+INIT:DONE,0
AT+VER

+VER:RA6W1-RRQ61001-24edf65ff2-1
OK
AT+SDKVER

+SDKVER:8.0.2.0.0
OK
```

Figure 18. RA6M4 USB console

Figure 19 shows the corresponding debug logs in the RA6M4 debug terminal, when sending `AT+VER` and `AT+SDKVER` commands from the RA6M4 console.

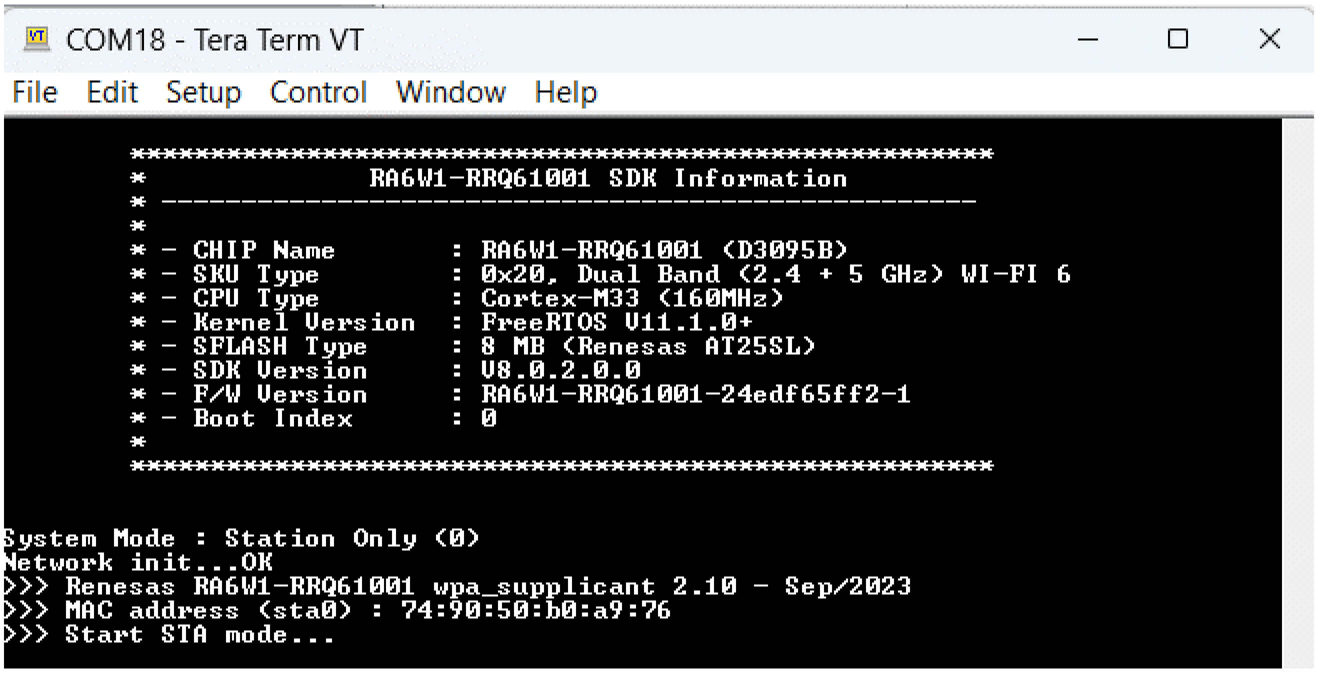


```
00> *   Renesas FSP Example Project for FreeRTOS_USB2SPI Module            *
00> *   Example Project Version 1.06                                  *
00> *   Flex Software Pack Version  6.0.0                            *
00> ******************************************************************
00> SystemCoreClock : 160000000
00>
00> SPITask start
00> ** SPI intr(irq10) open SUCCESSFUL **
00> ** SPI intr(irq10) enable SUCCESSFUL **
00> -- Created xSPICreatedEventGroup
00>
00> UARTTask start
00> -- Created xCreatedEventGroup
00>
00> USBTask USB task start
00> -- Created xUSBCreatedEventGroup
00> -- length:16
00> -- length:16
00> -- length:16
00> -- length:16
```

```
LOG: [1][0]: E000E000 CID B105900D PID 000BBD21 DEVARCH 47702A04 DEVTYPE 00 Cortex-M33
LOG: [1][1]: E0001000 CID B105900D PID 000BBD21 DEVARCH 47701A02 DEVTYPE 00 DWT
LOG: [1][2]: E0002000 CID B105900D PID 000BBD21 DEVARCH 47701A03 DEVTYPE 00 FPB
LOG: [1][3]: E0000000 CID B105900D PID 000BBD21 DEVARCH 47701A01 DEVTYPE 43 ITM
LOG: [1][5]: E0041000 CID B105900D PID 002BBD21 DEVARCH 47724A13 DEVTYPE 13 ETM
LOG: [1][6]: E0042000 CID B105900D PID 000BBD21 DEVARCH 47701A14 DEVTYPE 14 CSS600-CTI
LOG: RTT Viewer connected.
```

Figure 19. RA6M4 debug console

Figure 20 shows the RA6W1 CLI console logs after RA6W1 sending `AT+VER` and `AT+SDKVER` commands from the RA6M4 console.



```
        ******************************************************
        *              RA6W1-RRQ61001 SDK Information
        * ---------------------------------------------------
        *
        * - CHIP Name        : RA6W1-RRQ61001 (D3095B)
        * - SKU Type         : 0x20, Dual Band (2.4 + 5 GHz) WI-FI 6
        * - CPU Type         : Cortex-M33 (160MHz)
        * - Kernel Version   : FreeRTOS V11.1.0+
        * - SFLASH Type      : 8 MB (Renesas AT25SL)
        * - SDK Version      : V8.0.2.0.0
        * - F/W Version      : RA6W1-RRQ61001-24edf65ff2-1
        * - Boot Index       : 0
        *
        ******************************************************


System Mode : Station Only (0)
Network init...OK
>>> Renesas RA6W1-RRQ61001 wpa_supplicant 2.10 - Sep/2023
>>> MAC address (sta0) : 74:90:50:b0:a9:76
>>> Start STA mode...
```

Figure 20. RA6W1 CLI console

6. SDIO Interface

Secure Digital Input Output (SDIO) is an extension of the Secure Digital (SD) card interface standard, designed to support devices that need input/output capabilities. SDIO supports high-speed data transfers and is commonly used to interface with wireless communication modules (Wi-Fi and Bluetooth) or other devices that require a simple, fast connection.

The SDIO interface uses the signals below for communication.

- CLK (Clock): provides the clock signal for synchronization between the host and the SDIO device.
- CMD (Command): carries commands between the host and the SDIO device, including operational instructions like read, write, or status queries.
- DAT0, DAT1, DAT2, DAT3 (Data Lines): these are the data lines used for transferring data between the host and the SDIO device. SDIO can use 1, 4, or 8-bit data widths, depending on the specific mode and configuration.

6.1 SDIO Protocol – Message format

The format of the messages sent/received to/from the external processor is the RA6Wx protocol format over SDIO physical interface. The format and the parameters included are outlined in Figure 21.

For more information about SDIO protocol, see SDIO specification from SD association. When using SDIO protocol of RA6Wx, payload length should be aligned in units of 4-byte length. When 4 bytes align, fill with 0x00.

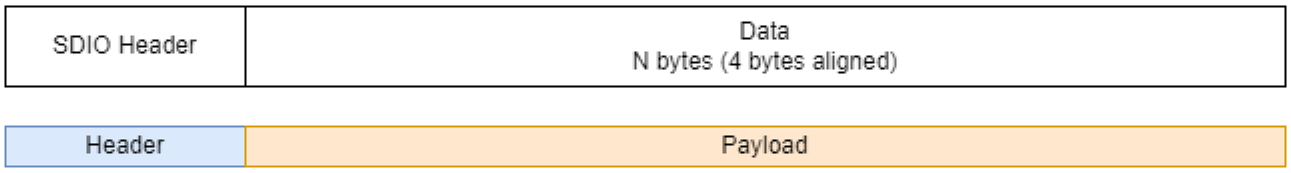


Figure 21. Basic format

6.2 AT Command -Sequence and Structure

6.2.1 SDIO Host to Device Sequence

SDIO host to device operations are performed in three SDIO transactions, see Figure 22.

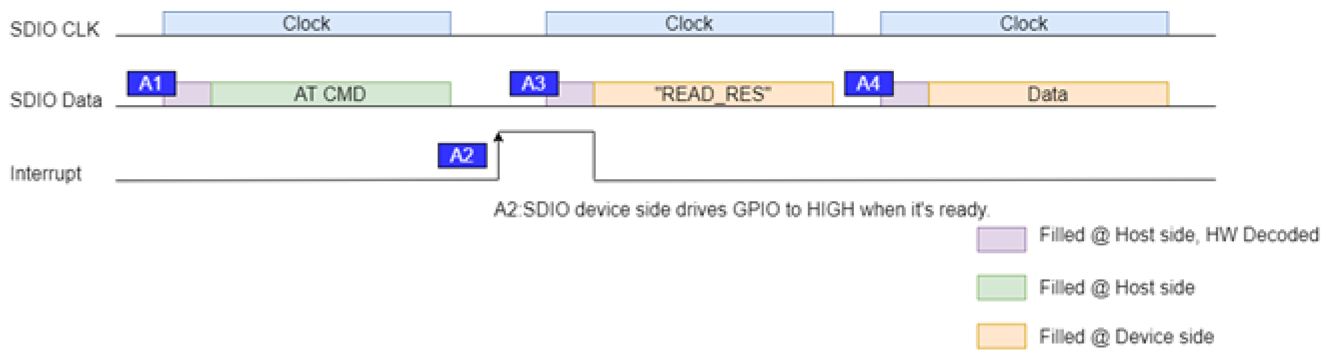


Figure 22. SDIO host to device sequence (AT command)

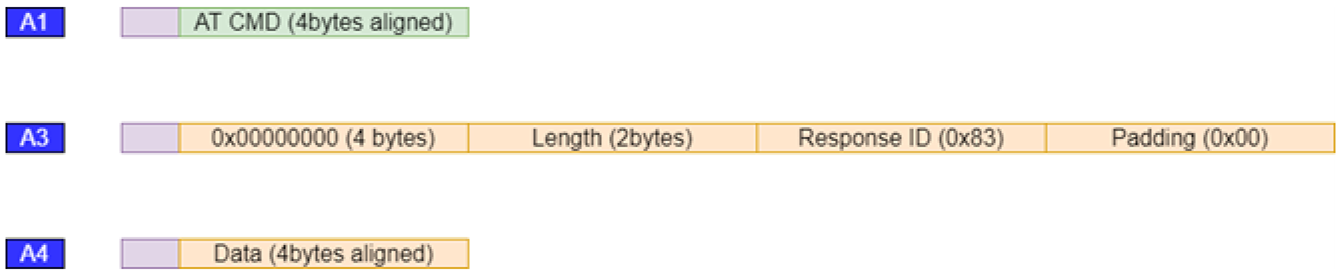


Figure 23. SDIO host to device data structure (AT command)

A1: The host sends an AT command to SDIO device.

A2: The host waits for GPIO interrupt line to go high.

A3: The host reads the response message from SDIO device. The response message format is shown below.

Table 7. Response message structure (AT command)

Offset	Length (B)	Contents	Value
0x00	4	Buffer address to read Address	Always 0x00000000.
0x04	2	Buffer Length	Data length of A4. This does not include the padding size when aligning to 4 B.
0x06	1	Response	Always 0x83.
0x07	1	Padding	Always 0x00.

A4: The host reads Data from SDIO device.

6.2.2 SDIO Device to Host Sequence

SDIO device to host operations are performed in two SDIO transactions.

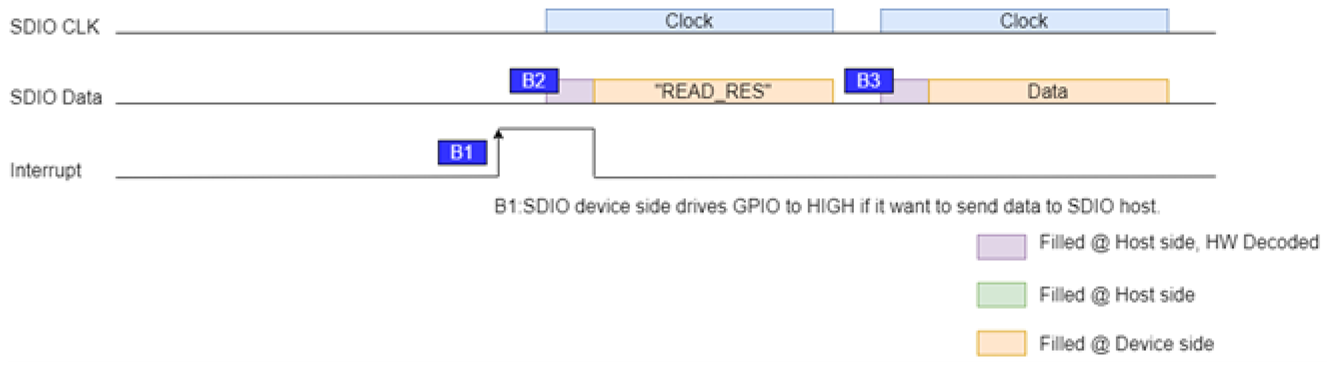

SDIO CLK
Clock
Clock
SDIO Data
B2
"READ_RES"
B3
Data
Interrupt
B1
B1:SDIO device side drives GPIO to HIGH if it want to send data to SDIO host.
Filled @ Host side, HW Decoded
Filled @ Host side
Filled @ Device side

Figure 24. SDIO device to host sequence (AT command)

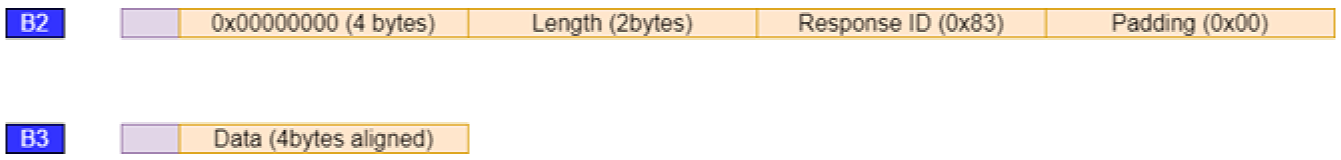


Figure 25. SDIO device to host data structure (AT command)

B1: The device triggers interrupt to inform the host when data is available.

B2: The host reads the response message from SDIO device. The response message format is same as Table 7.

B3: The host reads Data from SDIO device.

6.3 ESC Command -Sequence and Structure

6.3.1 SDIO Host to Device Sequence

The smallest unit of SDIO host to device operations are performed in three SDIO transactions.

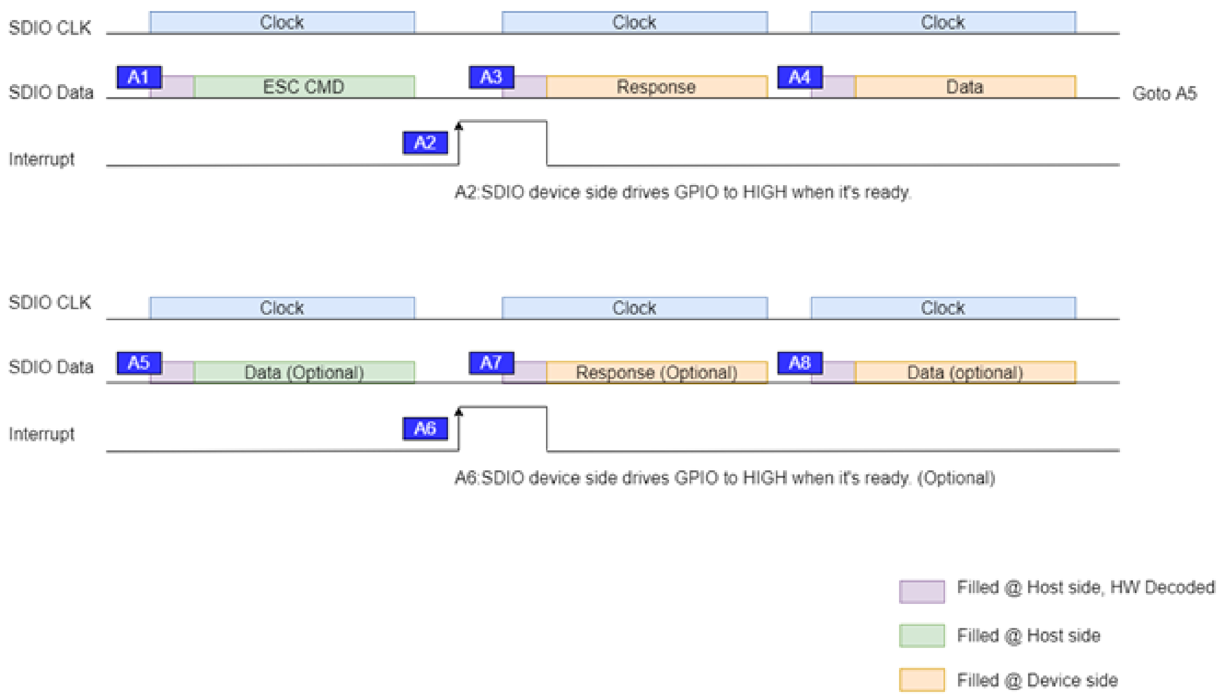


Figure 26. SDIO host to device sequence (ESC command)

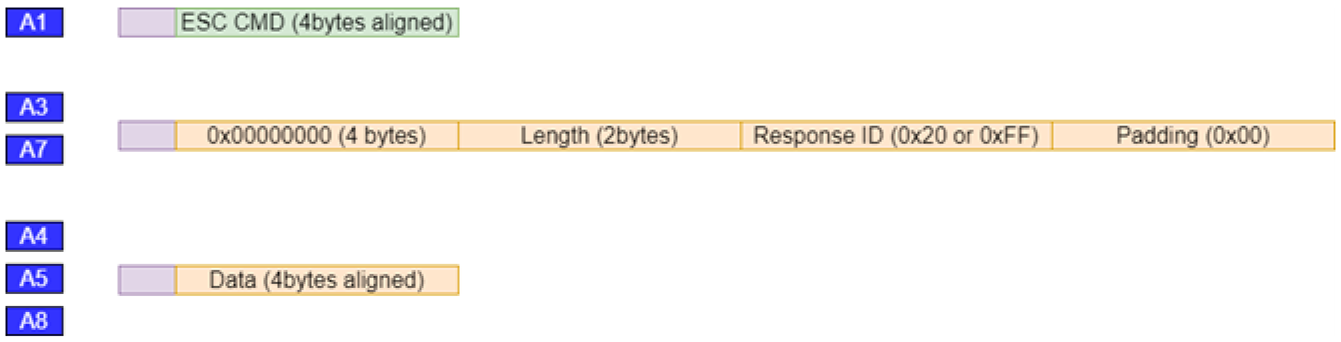


Figure 27. SDIO host to device data structure (ESC command)

A1: The host sends an ESC command to SDIO device.

A2: The host waits for GPIO interrupt line to go high.

A3: The host reads the response message from SDIO device. The response message format is shown in Table 8.

Table 8. Response message structure (ESC command)

Offset	Length (B)	Contents	Value
0x00	4	Buffer address to read Address	Accept ESC command 0x00000000 Cannot accept ESC command 0xFFFFFFFF
0x04	2	Buffer Length	Data length of A4. This does not include the padding size when aligning to 4 B.
0x06	1	Response	OK : 0x20 FAIL : 0xFF
0x07	1	Padding	Always 0x00.

A4: The host reads Data from SDIO device. For the <ESC> H command, the host sends data after A4 SDIO transaction.

A5: The host sends data to SDIO device.

A6: The host waits for GPIO interrupt line to go high.

A7: The host reads the response message from SDIO device.

A8: The host reads Data from SDIO device.

The response message format is same as in Table 8.

6.3.2 SDIO Device to Post lequence

There is no ESC command sequence from device to host. If it is necessary to send data from device to host, follow the procedure in Section 6.2.2 SDIO Device to Host Sequence

6.4 SDIO Pin MUX Configuration

Table 9 shows the SDIO interface pin configurations required in the RA6W1.

Table 9. Pin MUX configuration – SDIO

Usage	RA6Wx EVB
CLK	P0_08
CMD	P0_09
D0	P0_10
D1	P0_11
D2	P0_12
D3	P0_13
IRQ	P0_04

Whenever the RA6W1 has some data to be sent, it sends an interrupt to notify RA6M4.

- SDIO connector
  Use J220 to connect the host MCU (EK-RA6M4) and the RA6Wx EVB.

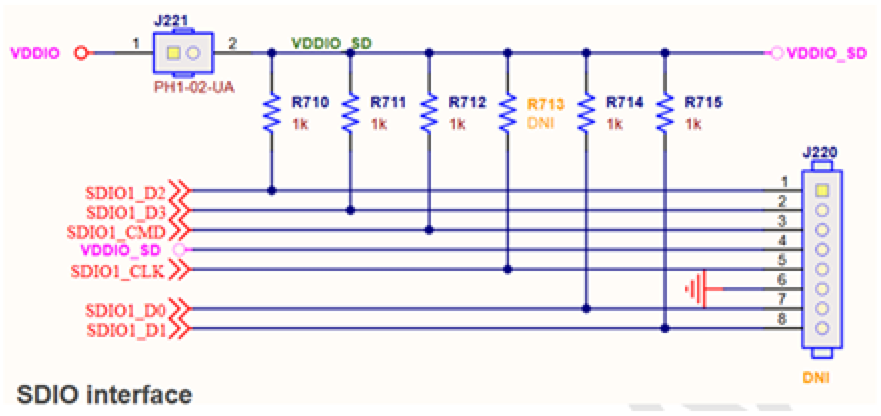


Figure 28. SDIO connector on RA6Wx EVB

To enable pull-up, check if the jumper configuration is as follows:

- Short J221
- Short #2 - #3 of J105, J106, J107

The signal line shown in Figure 28 is connected to the J203 on the RA6Wx EVB. Table 10 shows required connection between J201 and J203, and Figure 29 shows actual connection.

Table 10. Pin connection of J201 and J203 in RA6Wx EVB for SDIO Interface

J201	J203
P0_08 (PIN#18)	SDIO1_CLK (PIN#12)
P0_09 (PIN#20)	SDIO1_CMD (PIN#10)
P0_10 (PIN#22)	SDIO1_D0 (PIN#14)
P0_11 (PIN#24)	SDIO1_D1 (PIN#16)
P0_12 (PIN#26)	SDIO1_D2 (PIN#18)
P0_13 (PIN#30)	SDIO1_D3 (PIN#20)

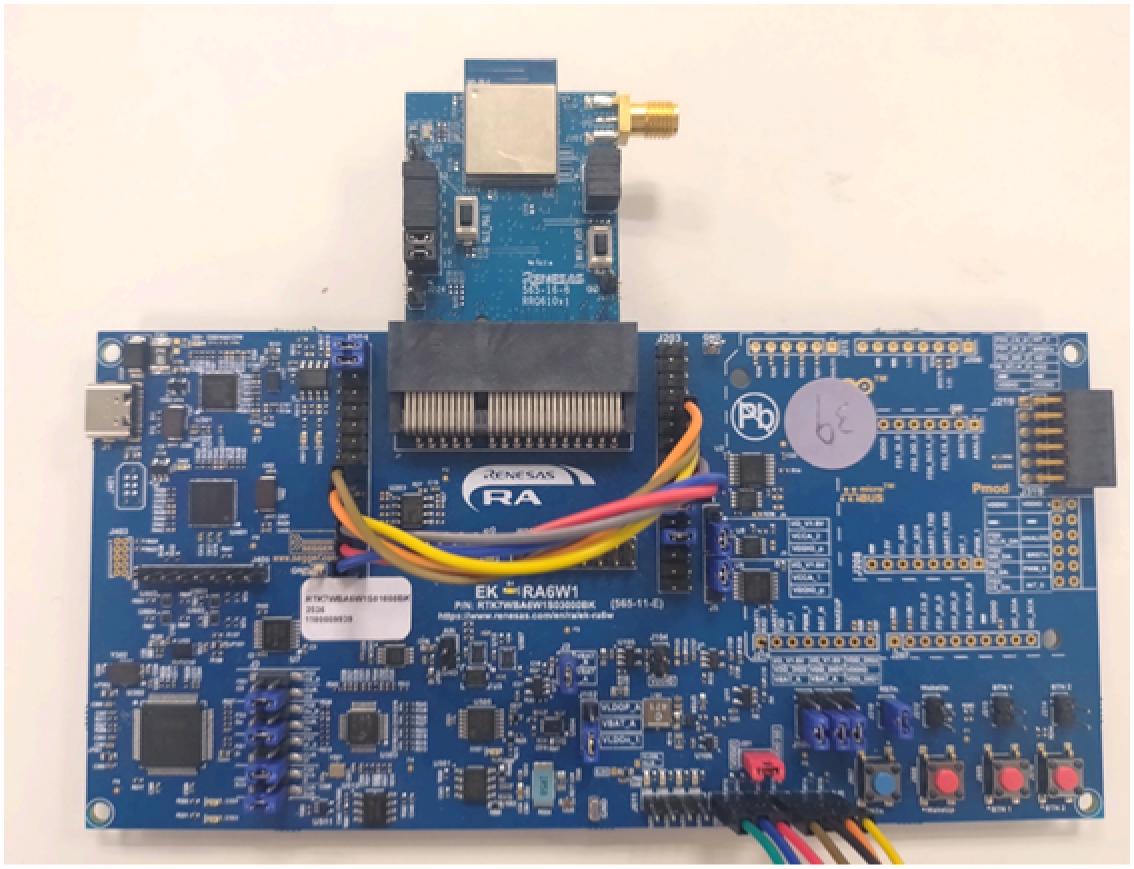


Figure 29. Pin connection of J201 and J203 in RA6Wx EVB for SDIO interface

The other required connection/configuration:

- Connect P0_04 on J201 to pin of host MCU that accepts external IRQ.
- The following jumper connections are also necessary
  - Connect FD3_CS on J305 to FD3_CS_D on J305
  - Connect FD2_DO on J304 to FD2_DO_D on J304
  - Connect FD1_DI on J303 to FD1_DI_D on J303
  - Connect FD0_SCLK on J302 to FD_SCLK_D on J302

6.5 Command over SDIO Example

An RA6M4 example program for AT command over SDIO can be downloaded from the RA6W1 example repository.

6.5.1 Overview

This example works by taking the user input AT command through the USB console of the RA6M4, bridges it from to SDIO of RA6M4 and sends the command to SDIO of the RA6W1 and gets the response to the console.

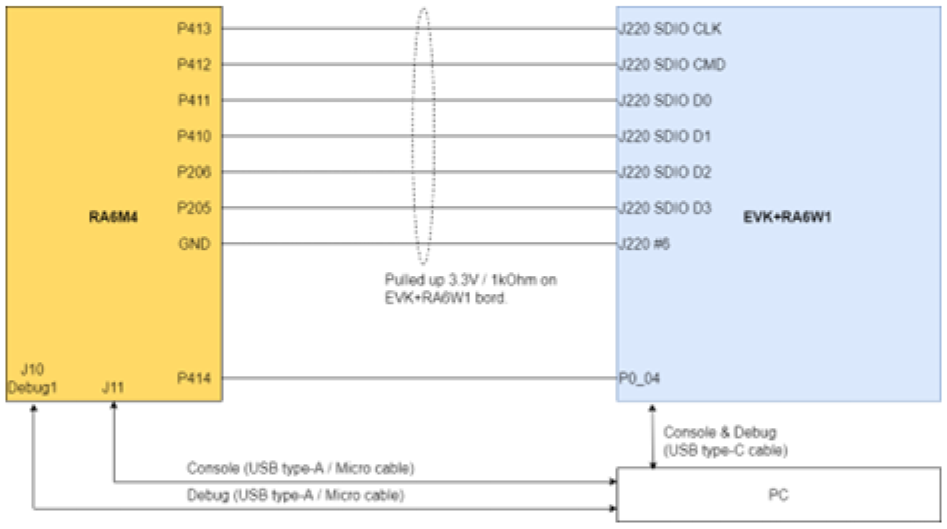


Figure 30. RA6M4+RA6W1 SDIO setup diagram

Table 11 shows pin interconnection required for interfacing AT command through SPI of EK-RA6M4 (MCU) with the RA6W1.

Table 11. Pin connection between EVK+RA6W1 and EK-RA6M4 for SDIO Interface

Usage	EK-RA6M4	RA6Wx EVB
CLK	P413 (J1 PIN #33)	SDIO_CLK (J220 PIN #5)
CMD	P412 (J1 PIN #34)	SDIO_CMD (J220 PIN #3)
Data[0]	P411 (J1 PIN #35)	SDIO_D0 (J220 PIN #8)
Data[1]	P410 (J1 PIN #36)	SDIO_D1 (J220 PIN #7)
Data[2]	P206 (J4 PIN #10)	SDIO_D2 (J220 PIN #1)
Data[3]	P205 (J4 PIN #11)	SDIO_D3 (J220 PIN #3)
GND	GND (Can be used any PINs silk -printed as “GND” on J1, J2, J3, J4.)	GND (J220 PIN #6)
IRQ	P414 (J1 PIN #32)	P0_04 (J201 PIN#10)

6.5.2 Components

- **RA6M4 USB console** (Type A micro cable):
  It is used to power RA6M4. The USB console can be accessed using Tera Term or similar applications. Table 12 shows required serial port setting. This console is used to send the AT commands to RA6M4.

Table 12. Serial Pin connection between EVK+RA6W1 and EK-RA6M4 for SDIO Interface

Item	Value
Baud rate	115200 [bps]
Data	8 [bit]
Parity	None
Stop bits	1 [bit]
Flow Control	None
New line (Receive)	AUTO
New line (Transmit)	CR+LF

- **RA6M4 Debug console**(Type A – micro cable):
  The debug port (DEBUG1) of RA6M4 is used for loading images using e^2 studio.
- **RA6W1 CLI console** (Type-C cable):

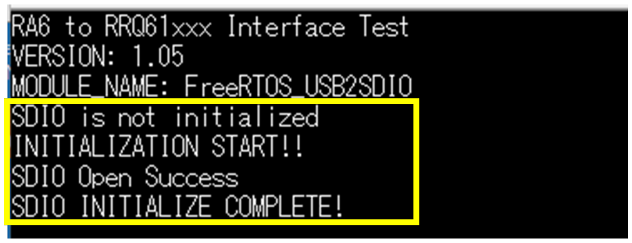
It is used to powerRA6W1, access the CLI console to view logs, and debugger. When connect EVB+RA6W1 to PC, two consecutively numbered COM ports will be added. The CLI console is assigned to the COM port with the lowest number. See Table 12 for required serial port configuration.

6.5.3 Working Example

This section describes the steps to program firmware image to EK-RA6M4 and RA6W1 to accept AT commands, and simple way to check if AT command works.

1. Load a program on EK-RA6M4 and EVB+RA6W1, for details see Ref. [3].
2. Launch terminal emulator (Teraterm) and open serial communications.
3. Reset the host (RA6M4) first followed by RA6W1 reset by pressing S1 (RA6M4), J607 (RA6W1).

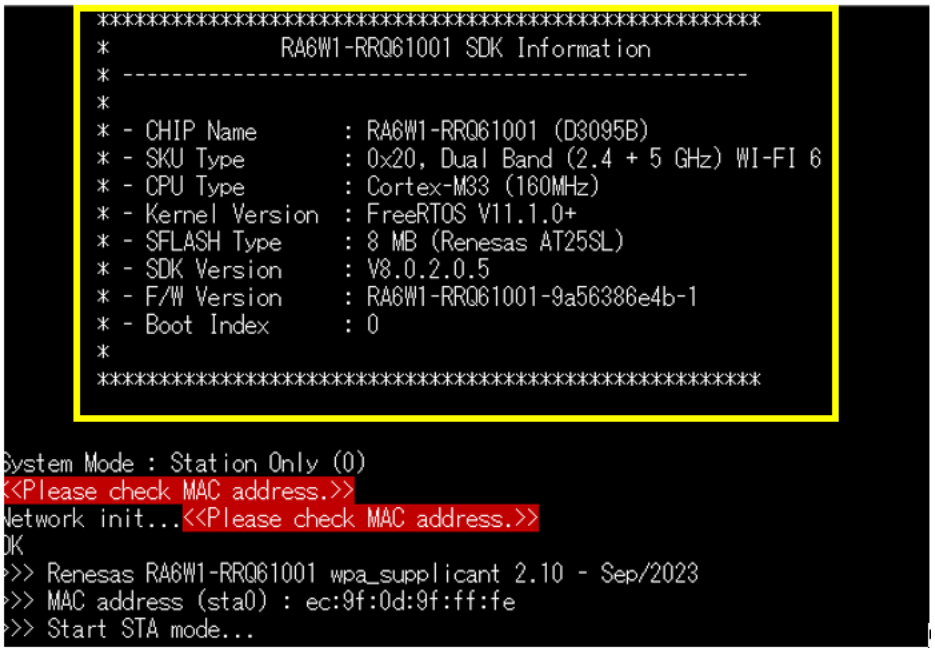
After RA6M4 andRA6W1 are reset, the messages appear in RA's console, see Figure 31.



```
RA6 to RRQ61xxx Interface Test
VERSION: 1.05
MODULE_NAME: FreeRTOS_USB2SDIO
SDIO is not initialized
INITIALIZATION START!!
SDIO Open Success
SDIO INITIALIZE COMPLETE!
```

Figure 31. SDIO initialization (RA6M4)

Figure 32 shows the CLI console of the RA6W1. Tthe message indicateing the operating environment for RA6W1 is in yellow rectangle.

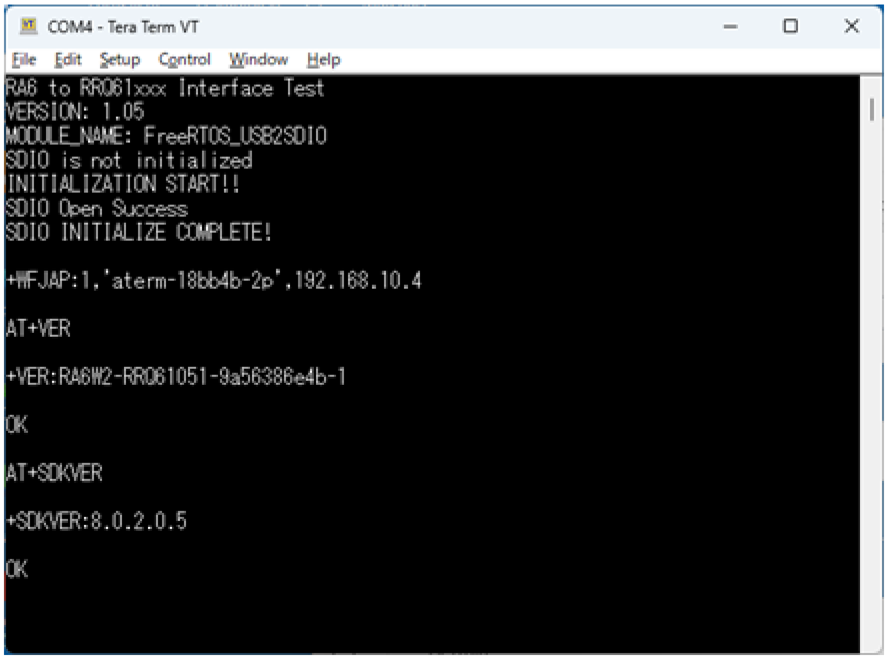


```
*******************************************************
*              RA6W1-RRQ61001 SDK Information
* ---------------------------------------------------
*
* - CHIP Name        : RA6W1-RRQ61001 (D3095B)
* - SKU Type         : 0x20, Dual Band (2.4 + 5 GHz) WI-FI 6
* - CPU Type         : Cortex-M33 (160MHz)
* - Kernel Version   : FreeRTOS V11.1.0+
* - SFLASH Type      : 8 MB (Renesas AT25SL)
* - SDK Version      : V8.0.2.0.5
* - F/W Version      : RA6W1-RRQ61001-9a56386e4b-1
* - Boot Index       : 0
*
*******************************************************

System Mode : Station Only (0)
<<Please check MAC address.>>
Network init...<<Please check MAC address.>>
OK
>> Renesas RA6W1-RRQ61001 wpa_supplicant 2.10 - Sep/2023
>> MAC address (sta0) : ec:9f:0d:9f:ff:fe
>> Start STA mode...
```

Figure 32. RA6W1 CLI console

Figure 33 shows sending `AT+VER` and `AT+SDKVER` command from RA6M4:

- If there is response which shown in Figure 33, EK-RA6M4 and RA6W1can accept AT command.
- if there is no response from RA6W1 to these AT commands, retry the step 3.



```
RA6 to RRQ61xxx Interface Test
VERSION: 1.05
MODULE_NAME: FreeRTOS_USB2SDIO
SDIO is not initialized
INITIALIZATION START!!
SDIO Open Success
SDIO INITIALIZE COMPLETE!

+WFJAP:1,'aterm-18bb4b-2p',192.168.10.4

AT+VER

+VER:RA6W2-RRQ61051-9a56386e4b-1

OK

AT+SDKVER

+SDKVER:8.0.2.0.5

OK
```

Figure 33. RA6M4 USB console

7. Revision History

Revision	Date	Description
1.03	Dec 5, 2025	Updated to Rev E Mother board Version. Updated SPI and SDIO section.
1.02	Aug 31, 2025	■ Updated AT UART to add RTS/CTS pins. ■ AT SPI interrupt pin change. ■ Updated images containing old logs.
1.01	May 30, 2025	■ Updated jumper connections in Section 4.2 RA6M4 + RA6W1 AT Command over UART Example and Section 6.4 SDIO Pin MUX Configuration. ■ Changed the chip name from RRQ61000 to RA6W1.
1.00	Mar 13, 2025	First release.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Renesas Electronics' suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

CorporateHeadquarters

TOYOSU FORESIA, 3-2-24 Toyosu
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact Information
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/